

2-D ANALYSIS OF U-FRAME STRUCTURES IN ELASTIC-PLASTIC SOIL MEDIUM

BY

HSI CHI YANG

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1988

ACKNOWLEDGEMENTS

The author wishes to express his deepest appreciation to his advisor, Dr. C. O. Hays for guiding him to nonlinear analysis. The author would also like to thank Dr. F. C. Townsend and Dr. L. E. Malvern for establishing his background on nonlinear soil behavior and continuum mechanics. The author is also grateful to Dr. M. C. McVay for his valuable guidance on constitutive models and geometric nonlinear analysis, and to Dr. F. E. Fagundo for his direct or indirect assistance in this work. Special thanks are given to Dr. M. E. Hoit for his unending assistance with the computer hardware and software.

Words are insufficient to express the author's gratitude to his parents, to whom this work is dedicated, for their encouragement and continuous support during all his education. Above all, he wishes to express his sincere appreciation and love to his wife Chen Ye-Chien for her patience and moral support during his graduate studies, and to his son, Louis, for understanding why they couldn't have much fun together while this work was being finished.

TABLE OF CONTENTS

	Page
ACKNOWLEDGEMENTS	ii
ABSTRACT	vi
CHAPTER 1. INTRODUCTION	1
Nonlinear Elastic Finite Element Analysis	1
Elastic-Plastic Finite Element Analysis	2
Geometrically Nonlinear Finite Element Analysis	3
Purpose of This Research	3
Outline of Presentation	4
CHAPTER 2. NONLINEAR ANALYSIS	6
Linear Finite Element Analysis	6
Nonlinear Finite Element Analysis	7
Incremental Step-by-Step Solution	10
Sequence-by-Sequence Solution	13
CHAPTER 3. UPDATED LAGRANGIAN FORMULATION	17
Formulation of the Continuum Mechanics Incremental Equations	17
Finite Element Discretization	24
Updated Lagrangian Jaumann Stress Rate Formulation	27
Limitations of the Jaumann Stress Rate Tensor	30
CHAPTER 4. CONSTITUTIVE LAWS	32
Linear Elastic Model	32
Variable Moduli Model	34
Plasticity Models	37
Interface Elements	50
CHAPTER 5. SIMULATION OF CONSTRUCTION SEQUENCES	53
In situ Stress	53

Dewatering	54
Excavation	55
Embankment	59
Adding and Deleting Bars	60
Reestablishment of the Water Table	60
Concentrated Loading	61
CHAPTER 6. COMPUTER PROGRAM	62
FEMCON -- An Overview	62
Overall Program Logic	63
Solution of Nonlinear Equations	68
Loading and Unloading Solution	69
CHAPTER 7. SOME ANALYTICAL VERIFICATIONS	74
Plane-Strain Elastic Large Deformation Problem	74
Settlement and Collapse Calculations of Footings	75
Uniaxial Volumetric Compaction	82
Test of Mixed Stiffness Method in Unloading	83
One Dimensional Excavation	85
Uniqueness Test for Excavation	88
Construction Sequences	93
CHAPTER 8. DISCRETE ELEMENT METHOD	98
Soil Response Curves by Finite Element Method	98
Analysis of U-frame Structures	102
CHAPTER 9. FIELD PROBLEMS	115
Port Allen Lock	115
Braced Excavation in Soft Clay	124
CHAPTER 10. CONCLUSIONS	147
Summary	147
Conclusions	148
Recommendations for Future Study	149
REFERENCES	151

APPENDICES	158
A MOHR-COULOMB ELASTIC-PLASTIC CONSTITUTIVE MATRIX	158
B CAP MODEL CONSTITUTIVE MATRIX	161
C INPUT GUIDE	166
D FORTRAN LISTING OF COMPUTER PROGRAM FEMCON	180
BIOGRAPHICAL SKETCH	292

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

2-D ANALYSIS OF U-FRAME STRUCTURES IN ELASTIC-PLASTIC SOIL MEDIUM

by

Hsi Chi Yang

August 1988

Chairman: Dr. Clifford O. Hays, Jr.
Major Department: Civil Engineering

A finite element analysis which considers geometric nonlinearity, material nonlinearity, and simulation of construction sequences is developed. The analysis was first developed to solve U-frame structures such as channels, basins and locks in soil medium. However, it can be used to solve general soil-structure interaction problems. The construction sequences include initial condition, dewatering, excavation, embankment, re-establishment of water level, and installation or removal of supports idealized as bar elements. A computer program has been written to implement and verify the analysis.

The method of analysis is based on the incremental method. The incremental method requires that the prescribed loads be applied in small but finite increments. The complete incremental solution can be obtained by using iterative techniques such as the tangent stiffness method, the constant stiffness method, and the modified tangent stiffness method. However, for problems with construction sequences, difficulties may arise for the tangent stiffness method and, therefore, a mixed stiffness method which can detect unloading and converges as fast as the tangent stiffness method is proposed.

In the nonlinear geometric analysis, the updated Lagrangian formulation is used. In the nonlinear material analysis, the elastic-plastic Mohr-Columb, Drucker-Prager

and Cap models are used as constitutive laws. The Drucker-Prager model is also used to verify the Rankine theory numerically.

The computer program is verified by solving 1) a number of problems involving various sequences of construction and problems involving nonlinear geometric and nonlinear material behavior and 2) complex field problems. The latter includes the analysis of a U-frame lock and the analysis of an open excavation involving a braced retaining wall. The predictions are compared with closed-form solutions, numerical results from other investigators, and field observations. It is felt that the finite element procedure and program developed can provide a general solution scheme for two-dimensional soil-structure interaction problems.

CHAPTER 1

INTRODUCTION

U-frame type structures such as channels, basins and locks are often used to facilitate flood control or navigation in inland waterways. To design these structures properly, the distribution of pressure on the base slab and earth pressure along the wall needs to be known accurately. These distributions are difficult to predict by using conventional procedures, principally due to the complexity of the interaction between the structure and the soil mass. If the complete loading history is considered, additional complications arise. The deformation behavior of soil media is usually dependent on the history of loading and unloading. Hence, it is more realistic to analyze U-frame structures by simulating the construction sequences in order to account for the stress history effects. In this research the finite element method will be used to analyze U-frame structures.

The direct stiffness method of finite element analysis is by now well established and widely used for the solution of small displacement, elastic structural or geotechnical problems. Various types of elements have been developed and general purpose computer programs now exist. Since the establishment of the method, there has been much interest in extending the method for nonlinear analysis. The nonlinearities arise from two distinct sources: material nonlinearities and geometric nonlinearities, the latter being due to large displacements.

Nonlinear Elastic Finite Element Analysis

Clough and Duncan [17, 36] developed a very elaborate finite element program called SOIL-STRUCT to analyze the Port Allen Lock. The program used a nonlinear

elastic model by Duncan and Chang [35] to describe the soil stress-strain behavior and has subroutines for stress initialization and simulating incremental construction and loading. The incremental construction and loading may consist of excavation and dewatering of the soil, buildup of the structure and backfill, reestablishment of normal ground water conditions, boundary water pressure loading on the structure, and temperature loading of structural material. SOIL-STRUCT also has special interface elements to allow the relative displacements on the boundary between the structure and the surrounding soil mass. The behavior of these interface elements can again be expressed by the nonlinear elastic model. Radhakrishnan and Jones [75] later did some minor modifications to SOIL-STRUCT. The modified program still uses the same nonlinear elastic soil model.

Elastic-Plastic Finite Element Analysis

Although nonlinear elastic models have been applied to soils leading to useful results, the main physical feature leading to a soil's nonlinear behavior is the irrecoverability of strain. A convenient mathematical framework for describing this phenomenon is to be found in the theory of plasticity. In this research, elastic-plastic models based on the theory of plasticity will be used to describe the soil stress-strain behavior. Many plasticity models have been proposed [1, 31, 32, 58, 59, 79, 82]. Accurate modeling of soil is very difficult because of the various parameters involved, which determine the behavior of the model. But it is still possible to come up with a reasonable accurate model for particular cases.

Many finite element analyses have been done by using elastic-plastic material models and some of them include construction sequences [24, 27, 61, 64, 81]. Mana [64] used the Drucker-Prager model in the finite element analyses of deep excavation behavior in soft clay. The same model was used by Sargand [81] in a hybrid finite element procedure for soil-structure interaction and construction sequences. Lightner

[61] continued Sargand's work by using Von Mises, Drucker-Prager and Cap models in a mixed finite element analysis.

Geometrically Nonlinear Finite Element Analysis

For a number of years, the basic tools to treat structural and geotechnical problems have been small-strain elasticity and plasticity. Significantly less effort has been spent to introduce the equally important concept of finite strains and large deformations. Although in structural engineering the geometric nonlinearity has been applied to plate, shell and beam-column structures [5, 45, 46, 54, 70], small displacement and small strain assumptions still seem to be acceptable for many steel and concrete structures. However, this assumption seems to have very limited applicability in geotechnical engineering. The soil medium is generally subjected to loadings which give rise to large deformations. The area around a penetrating pile is a very good example of the inadequacy of small strain theory to describe the actual physical phenomena. The soil medium deformation close to a shallow footing is another good example of a problem exhibiting strong geometrical nonlinearities. It would be more realistic to consider the geometric nonlinearity in soil-structure interaction problems.

Purpose of This Research

Investigation of structures involving material and geometric nonlinear behavior has been a subject of study for a long time. Notable contributions have been made by various investigators using the finite element method. Most of these studies have considered systems such as framed and space structures, plates and shells; some studies have considered nonlinear behavior of geological bodies. However, no computer analysis is known that considers the effects of both geometric and material nonlinearities in the simulation of construction sequences. The purpose of this research is to develop such a computer analysis.

The research described herein is an attempt to examine some of the features of a large deformation finite element formulation as applied to nonlinear material problems of modeling construction sequences. The research is divided into four main parts. They are

- (1) theoretical development
- (2) computer implementation of the theory
- (3) verification of the theory and computer code through
the solution of a variety of problems
- (4) analysis of U-frame structures

Outline of Presentation

Chapter 2 reviews a general method of nonlinear analysis called the incremental method which is well suited for the stress-dependent nonlinear geometric and/or material problems.

In Chapter 3, the incremental development of the governing continuum mechanics equations for elastic-plastic bodies subjected to large strains and large displacements is presented by using the updated Lagrangian formulation. Then the finite element method is used to solve these equations.

Chapter 4 gives a brief review of the theories of elasticity and plasticity, then describes the constitutive laws which are used in this research. Also, in this chapter the constitutive laws for two interface-element models are presented.

Chapter 5 describes the procedures for modeling various stages of construction sequences such as insitu stresses, dewatering, excavation, embankments and support systems.

In Chapter 6, a brief description of the associated computer program is given. After that, the iteration methods used in the program to solve the nonlinear incremental

equations are discussed and a new iteration method is proposed to solve problems with unloading.

In Chapter 7, the computer program is verified by solving a few available problems and by comparing predictions with the exact solutions or the solutions obtained by other investigators.

In Chapter 8, the discrete element method is applied to solve the U-frame structure. The q-w (soil response) curves used in the discrete element analysis are generated using the finite element method.

Chapter 9 presents the practical application of the new computer analysis to two field structures involving soil-structure interaction with construction sequences. The obtained results are compared with those from observed field data.

Finally, in Chapter 10, a discussion of the completed research is given with recommendations for future studies.

CHAPTER 2

NONLINEAR ANALYSIS

In this chapter, an extremely powerful method of nonlinear analysis called the incremental method is reviewed. The incremental method requires that the prescribed loads be applied in small but finite increments. The complete incremental solution is shown to contain an iterative solution within each increment. The incremental solution algorithm is then extended to problems with construction sequences.

Linear Finite Element Analysis

In the linear finite element analysis, it is assumed that the displacements and strains developed in the structure are infinitesimally small and that the material is linearly elastic. With these assumptions, the finite element equilibrium equations can be written for static analysis as [4, 77]

$$[K]\{U\} = \{Q\} \quad (2.1)$$

$$\text{where } \{Q\} = \{Q\}_B + \{Q\}_S + \{Q\}_C \quad (2.2)$$

The load vector $\{Q\}$ includes the effect of the element body forces,

$$\{Q\}_B = \sum_m \int_V [N](m)^T \{f_b\}(m) dV(m) \quad (2.3)$$

the effect of the element surface forces,

$$\{Q\}_S = \sum_m \int_V [N]^{(m)T} \{f_s\}^{(m)} dV^{(m)} \quad (2.4)$$

and the effect of the concentrated loads, $\{Q\}_C$.

The matrix $[K]$ is the stiffness matrix of the element assemblage,

$$[K] = \sum_m \int_V [B]^{(m)T} [C]^{(m)} [B]^{(m)} dV^{(m)} \quad (2.5)$$

In the above equations, $[B]^{(m)}$ is the strain-displacement matrix, $[C]^{(m)}$ is the elasticity matrix, $[N]^{(m)}$ is the displacement interpolation matrix, $\{f_b\}^{(m)}$ is the vector of body forces, $\{f_s\}^{(m)}$ is the vector of surface tractions and the superscript m denotes element m .

These equations correspond to a linear analysis because the displacement response $\{U\}$ is a linear function of the applied load vector $\{Q\}$; i.e., if the loads are $n\{Q\}$ instead of $\{Q\}$, where n is a constant, the corresponding displacements are $n\{U\}$. However, this is not the case for a nonlinear analysis.

The linearity of a response prediction rests on the assumptions stated above and it is instructive to identify where these assumptions have entered into the finite element equilibrium equations. The fact that the displacements must be small has entered into the evaluation of the matrix $[K]$ and load vector $\{Q\}$, because all integrations have been performed over the original volume of the finite elements, and the strain-displacement matrix $[B]$ as well as the displacement interpolation matrix $[N]$ of each element was assumed to be constant and independent of the element displacements. The assumption of a linear elastic material is implied in the use of a constant stress-strain matrix $[C]$.

The above discussion of the basic assumptions used in a linear analysis implicitly defines the material and geometric nonlinearities to be considered in this study.

Nonlinear Finite Element Analysis

In the materially nonlinear analysis, the nonlinear effect lies in the nonlinear stress-strain relation, which is dependent on the value of the nodal displacements. The displacements and strains are assumed to be infinitesimally small. In the geometrically nonlinear analysis, the material is subjected to large displacements and large strains. In other words, the changes of geometry brought by displacements can no longer be neglected and the nonlinear strain-displacement relation is considered in the analysis. In this case the stress-strain relation is also usually nonlinear.

Before discussing the nonlinear analysis, a time variable t is first introduced to conveniently describe the loading and the motion of a body. In a static analysis without time effects (e.g., without creep effects), time is only a variable which denotes different intensities of load applications and corresponding different configurations.

The basic problem in a general nonlinear analysis is to find the state of equilibrium of a body corresponding to the applied loads. Whether the analysis is linear or nonlinear, equilibrium conditions between internal and external forces have to be satisfied. Thus, if the displacements are prescribed by a finite number of nodal displacements, the necessary equilibrium equations can be obtained by using the virtual work principle as

$${}^t\{Q\} - {}^t\{F\} = \{0\} \quad (2.6)$$

$$\text{with } {}^t\{F\} = \sum_m \int_{t_V(m)} {}^t[B](m)^T {}^t[\tau](m) {}^t dV(m) \quad (2.7)$$

where the vector ${}^t\{Q\}$ lists the externally applied nodal point forces in the configuration at time t , and the vector ${}^t\{F\}$ lists the nodal forces that correspond to the element

stresses $\{\tau\}$ in this configuration. In a general large deformation analysis the stress as well as the volume of the body at time t is unknown.

The discretized general nonlinear finite element equations can be obtained from Equations 2.6 and 2.7 and written in terms of the unknown nodal displacement as

$$^t[K(U)]^t\{U\} = ^t\{Q\} \quad (2.8)$$

While the solution of a linear equation system can be accomplished without any difficulty in a direct manner, this is not possible for nonlinear systems. The nonlinear solution process consists in searching vector $\{U\}$ that renders the residual, $\{Q\} - ^t[K(U)]^t\{U\}$, as small as possible. An exact solution would make the residual vanish. The above nonlinear equations are generally solved using the following iteration methods or methods derived from them:

- (1) secant method
- (2) Newton-Raphson method (tangent stiffness method)
- (3) incremental method

In some nonlinear static problems only the displacements and stresses reached at specific load levels are required. In such a case, the solutions can be accomplished in a one-step operation using the Secant or Newton-Raphson method. However, when the analysis includes path-dependent nonlinear geometric or material conditions, the equilibrium relations in Equation 2.6 need be solved for the entire history of loading. It would then appear that it is necessary to solve the nonlinear problem in steps. The calculation of the step by step solution can be effectively carried out by the incremental method, in which the prescribed loads are considered to be applied in small but finite increments. Before describing the incremental solution, some of the advantages of the incremental method are listed below:

(1) The incremental analysis reduces to a one-step analysis if in a static solution the total load is applied all together and only the configuration corresponding to that load is calculated.

(2) The solution of the one-step analysis is not guaranteed to converge in all circumstances. In practice, the analysis of such a case frequently requires an incremental solution with a number of load steps to finally reach the total applied load.

(3) With the incremental method, by choosing a suitably small load increment convergence can be guaranteed and, as pointed out by Zienkiewicz [97], results of a reasonable kind will generally be available.

In the next chapter, the incremental formulation is employed to derive the governing finite element equations used in this research.

Incremental Step-by-Step Solution

The basic approach in an incremental step-by-step solution is to assume that the solution for the discrete time t is known, and that the solution for discrete time $t+\Delta t$, which is the solution after adding a suitably chosen load increment, is required. The equilibrium equations at time $t+\Delta t$ can be expressed as

$$t+\Delta t\{Q\} - t+\Delta t\{F\} = \{0\} \quad (2.9)$$

Since the solution is known at time t , $t+\Delta t\{F\}$ can be written as

$$t+\Delta t\{F\} = t\{F\} + \{\Delta F\} \quad (2.10)$$

where $\{\Delta F\}$ is the vector of incremental nodal forces due to the increment in element stresses from time t to time $t+\Delta t$. This vector can be approximated using the following linearized equation:

$$\{\Delta F\} = {}^t[K]\{\Delta U\} \quad (2.11)$$

where ${}^t[K]$ is the tangent stiffness matrix corresponding to the geometric and material condition at time t and $\{\Delta U\}$ is a vector of incremental nodal point displacements. The incremental form of the discretized nonlinear equations can be obtained by substituting Equations 2.11 and 2.10 into Equation 2.9 as

$${}^t[K]\{\Delta U\} = {}^{t+\Delta t}[Q] - {}^t[F] \quad (2.12)$$

After solving Equation 2.12 for $\{\Delta U\}$, an approximation to the displacements at time $t + \Delta t$ can be calculated,

$${}^{t+\Delta t}[U] = {}^t[U] + \{\Delta U\} \quad (2.13)$$

The calculation in Equation 2.13 is only an approximation to the exact displacements at time $t + \Delta t$ because Equation 2.11 was used.

After having evaluated an approximation to the displacements corresponding to time $t + \Delta t$, an approximation to the stresses and corresponding nodal point forces at time $t + \Delta t$ can be solved and then the next increment calculations can be performed. However, because of the linearized incremental relations in Equation 2.11, such a solution may deviate significantly from the real nonlinear path and, depending on the load increment sizes used, may indeed be unstable. To correct this deviation, in practice, it is therefore frequently necessary to iterate until the solution of Equation 2.9 is obtained to sufficient accuracy.

If the Newton-Raphson iteration is used, Equations 2.12 and 2.13 can be written, for $i = 1, 2, 3, \dots$, as

$$t[K]\{ΔU\}^{(i)} = t[Q] - t[F]^{(i-1)} \quad (2.14)$$

$$t[Q]^{(i)} = t[Q]^{(i-1)} + \{ΔU\}^{(i)} \quad (2.15)$$

with the initial conditions $t[Q]^{(0)} = t[K]$, $t[U]^{(0)} = t[U]$, and $t[F]^{(0)} = t[F]$.

Instead of computing a different stiffness matrix for each iteration, some modified iterative techniques have been employed. One such modification is to use the initial stiffness matrix, $[K]$, at each iteration. Thus, Equation 2.14 can be modified as

$$[K]\{\Delta U\}^{(i)} = t[Q]^{(i-1)} - t[F]^{(i-1)} \quad (2.16)$$

with the initial conditions $t[F]^{(0)} = t[F]$, $t[U]^{(0)} = t[U]$. This "initial stiffness" method may result in a very slowly convergent solution.

Another modification of the tangent stiffness algorithm is that the tangent stiffness is revised only between load increments so that a reformation of the stiffness matrix between load increments is not required. In other words, the tangent stiffness matrix is updated at the beginning of each increment. Thus, Equation 2.14 can be written as

$$t[K]\{\Delta U\}^{(i)} = t[Q] - t[F]^{(i-1)} \quad (2.17)$$

with the initial conditions $t[F]^{(0)} = t[F]$, $t[U]^{(0)} = t[U]$. The above approach is probably best when a large number of increments are applied.

It was observed by many investigators [22, 25, 98] and by the author that usually the Newton-Raphson iteration is the fastest, most accurate, and most economical. The solution within an increment can generally converge with very high accuracy in less

than about ten iterations. However, the program documented herein allows the user to choose any one of the three iterations describe above.

The above iteration solution used might be described as an incremental loading iterative solution; that is, loads are applied in increments, and within each increment an iterative solution is performed until the full value of loading at the end of the increment is absorbed by the structure. Figures 2-1, 2-2, and 2-3 show respectively how each of the three iterations works with the incremental method in a one dimensional problem. It will be shown in Chapter 6 that a new iteration method is proposed to solve problems with unloading.

Sequence-by-Sequence Solution

To extend the above solution procedure to problems with construction sequences, the above incremental loading iterative solution must be performed for each construction sequence. The sequence-by-sequence solution might be described as a sequential incremental loading iterative solution; that is, loads are applied in sequences, within each sequence loads are applied in increments, and within each increment an iterative solution is performed. The details of forming the tangent stiffness matrix are developed in Chapter 3 and Chapter 4; and the details of forming the loading matrix for each construction sequence are discussed in Chapter 5.

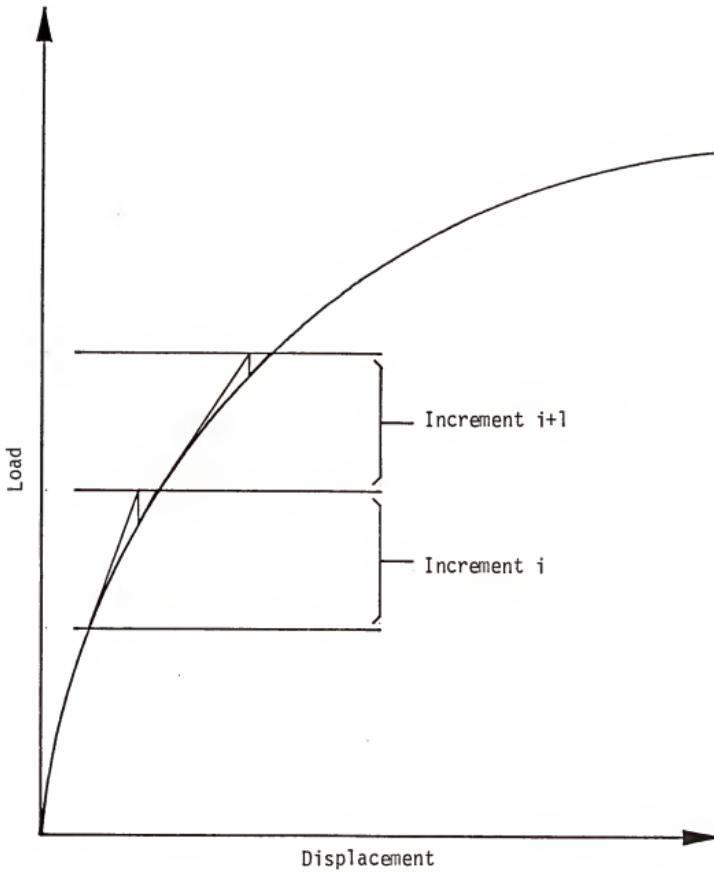


Figure 2-1 Tangent stiffness method

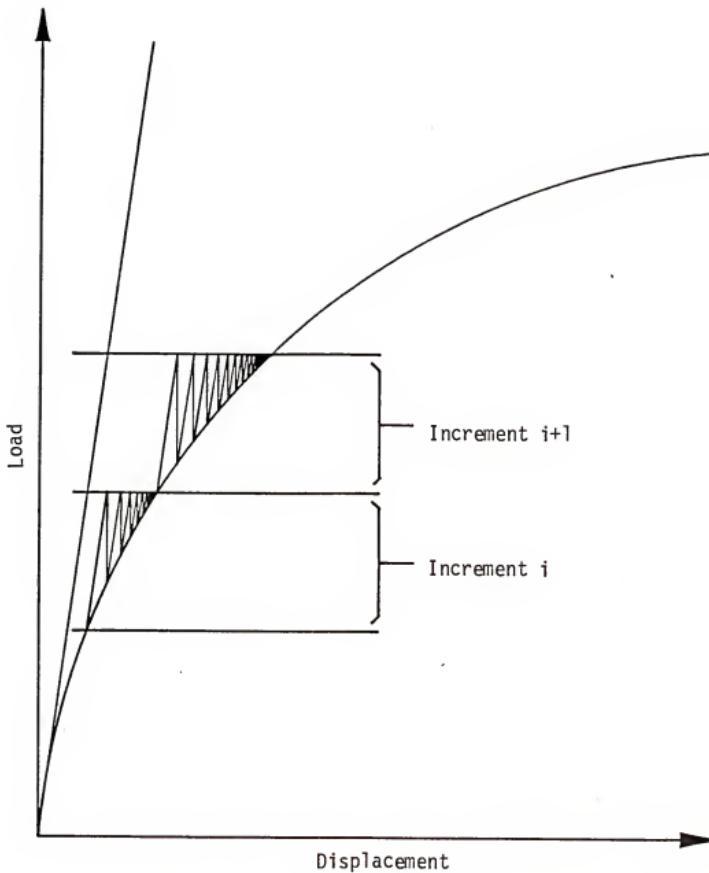


Figure 2-2 Constant stiffness method

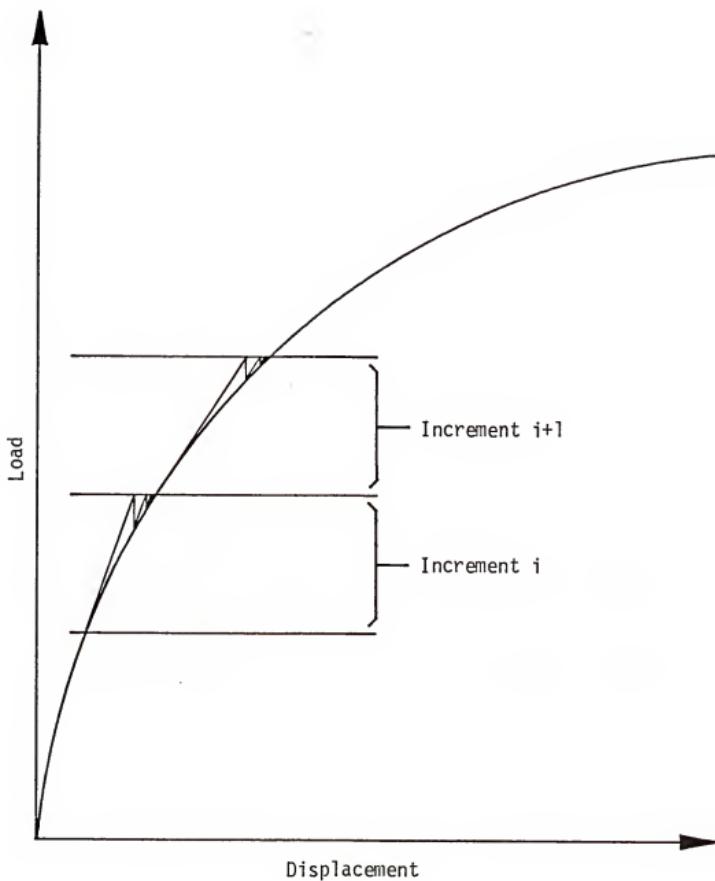


Figure 2-3 A modified stiffness method (Recompute the tangent stiffness at the begining of each increment)

CHAPTER 3

UPDATED LAGRANGIAN FORMULATION

As stated in Chapter 2, in nonlinear analyses dealing with both geometric and material nonlinearities, an incremental solution scheme is the most suitable approach. In this chapter, a general incremental formulation for elastic-plastic bodies subjected to large strains and large displacements is presented. The formulation is in updated Lagrangian coordinates and is based on the principle of virtual work. In the formulation, it is shown that the iterative methods can be applied to the obtained governing incremental continuum mechanics equations. Then the governing finite element equations used in this study are derived from these equations. In the incremental solution, the Jaumann stress-rate tensor is used to update the stresses in the body. However, a discussion of the Jaumann stress-rate tensor indicates that it can only apply to isotropic materials.

Formulation of the Continuum Mechanics Incremental Equations

In the development to follow, the motion of a general body is considered in a stationary Cartesian coordinate system, and it is assumed that the body can experience large displacements, large strains and a nonlinear stress-strain response. To describe the deformation of a body, as shown in Figure 3-1, three configurations at different times are introduced:

the initial state $t = 0$

the current state $t = t$

the subsequent state $t = t + \Delta t$

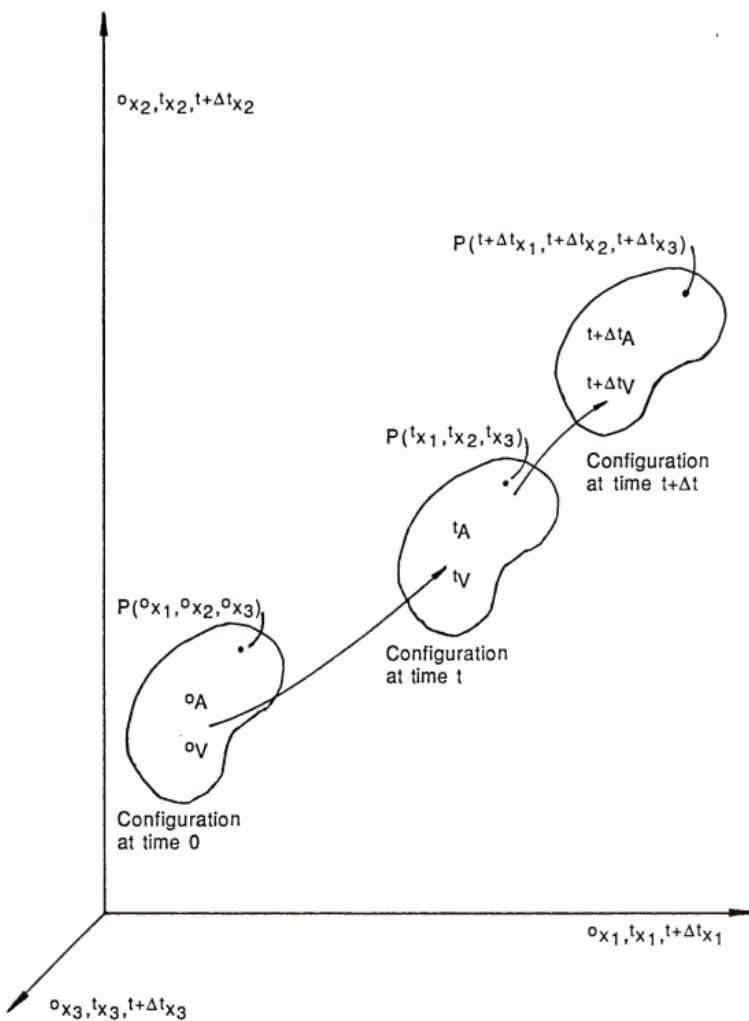


Figure 3-1 Motion of body in stationary cartesian coordinate system

To formulate the equations governing an increment of deformation, the current and subsequent configurations are used. The stresses, strains and displacements in the current configuration are presumed known and have been determined through a sequence of incremental steps. The subsequent configuration is reached through a further increment of deformation, and it is the incremental stresses, strains and displacements at time $t+\Delta t$ that have to be determined.

It is useful at this point to describe the notation which will be used. In presenting the continuum mechanics relation for the large deformation analysis, the traditional indicial and tensor notation is employed [38, 63]. In addition to that, a left subscript and superscript notation is introduced [4, 6]. A left superscript indicates in which configuration the variable or the quantity occurs. For example, the coordinates of a point in the body at time 0 are ${}^0x_1, {}^0x_2, {}^0x_3$; at time t they are ${}^t x_1, {}^t x_2, {}^t x_3$; at time $t+\Delta t$ they are ${}^{t+\Delta t} x_1, {}^{t+\Delta t} x_2, {}^{t+\Delta t} x_3$ and the volumes of the body at times 0, t , and $t+\Delta t$ are ${}^0V, {}^t V$, and ${}^{t+\Delta t} V$.

Since the configuration of the body at time $t+\Delta t$ is not known, as it will be shown later, it is necessary to refer applied forces, stresses, and strains to a known equilibrium configuration. In analogy to the notation used for coordinates, a left superscript indicates in which configuration the quantity (stress, strain, etc.) occurs; in addition, a left subscript indicates the configuration with respect to which the quantity is measured. For example, the body force components at time $t+\Delta t$, but measured in configuration 0, are ${}^{t+\Delta t} {}_0^fb_i$, $i = 1, 2, 3$.

Before beginning the formulation, it might be helpful to introduce some of the terminology to be used here [4, 38, 63]:

τ_{ij} -- Current Cartesian stress tensor referred to a global reference frame. It is to be clearly understood that these are the physical stress components representing force per unit area of the current geometry. This tensor is called Cauchy stress tensor in the engineering mechanics literature.

S_{ij} -- 2nd Piola-Kirchhoff stress tensor. This tensor will be formally defined later, it being noted however in this study the 2nd Piola-Kirchhoff stress tensor describes the stress state in the subsequent configuration as referred to the geometry of the current configuration. These are not physical stress components as defined above.

ϵ_{ij} -- Green-Lagrange strain tensor. This is the tensor used with the 2nd Piola-Kirchhoff stress tensor and is defined as

$$\epsilon_{ij} = \frac{1}{2} (u_{i,j} + u_{j,i} + u_{k,i}u_{k,j}) \quad (3.1)$$

e_{ij} -- Infinitesimal strain tensor. This tensor is defined as

$$e_{ij} = \frac{1}{2} (u_{i,j} + u_{j,i}) \quad (3.2)$$

Referring to Figure 3-1, equilibrium of the subsequent state is implied by the virtual work expression

$$\int_{t+\Delta t_V}^{t+\Delta t} \tau_{ij} \delta u_i \delta u_j dV = t+\Delta t W \quad (3.3)$$

The right-hand side of Equation 3.3 is the virtual work performed when the body is subjected to a virtual displacement. The corresponding external virtual work is W , with

$$t+\Delta t W = \int_{t+\Delta t_V}^{t+\Delta t} b_i \delta u_i t+\Delta t dV + \int_{t+\Delta t_S}^{t+\Delta t} s_i \delta u_i t+\Delta t dS \quad (3.4)$$

where the $t+\Delta t b_i$ and $t+\Delta t s_i$ are the components of the externally applied body and surface force vector, respectively. In Equations 3.3 and 3.4, δu_i is a (virtual) variation

in current displacement components $t+\Delta t u_i$, and $\delta_{t+\Delta t} \epsilon_{ij}$ are the corresponding (virtual) variation in strains; i.e.

$$\delta_{t+\Delta t} \epsilon_{ij} = \delta \frac{1}{2} (t+\Delta t u_{i,j} + t+\Delta t u_{j,i}) \quad (3.5)$$

Equation 3.3 expresses the equilibrium and requirement of the body considered in the subsequent configuration and the constitutive equations enter Equation 3.3 in the calculation of the stresses.

A fundamental difficulty in the general application of Equation 3.3 is that the subsequent configuration of the body at time $t+\Delta t$ is unknown. To solve Equation 3.3, it would appear to be necessary first to refer all the variables to a previously calculated configuration and then the resulting equations can be solved by using the linearization process described in Chapter 2. If they are referred to the undeformed (original) configuration, the approach is called the total Lagrangian formulation and if they are referred to the recently computed configuration, the approach is called the updated Lagrangian formulation [3, 4, 6, 25, 49, 63, 72]. In this study, all variables at time $t+\Delta t$ are referred to the current configuration at time t . Thus, the updated Lagrangian formulation is used to develop the governing incremental equations.

In the updated Lagrangian formulation, Equation 3.3 is transformed into

$$\int_{t_V}^{t+\Delta t} S_{ij} \delta^{t+\Delta t} \epsilon_{ij} t dV = t+\Delta t W \quad (3.6)$$

where $t+\Delta t S_{ij}$ are the second Piola-Kirchhoff stresses, related to the Cauchy stresses by

$$t+\Delta t S_{ij} = \frac{t_p}{t+\Delta t p} t+\Delta t x_{i,m} t+\Delta t \tau_{mn} t+\Delta t x_{j,n} \quad (3.7)$$

and ε_{ij} are the Green-Lagrange strains.

By using an incremental decomposition, the stresses and strains at time $t + \Delta t$ can be written as

$${}^{t+\Delta t}S_{ij} = {}^t\tau_{ij} + S_{ij} \quad (\text{note that } {}^tS_{ij} = {}^t\tau_{ij}) \quad (3.8)$$

$$\delta {}^{t+\Delta t}e_{ij} = \delta {}_t e_{ij} + \delta {}_t n_{ij} \quad (3.9a)$$

$$\text{with } {}_t e_{ij} = \frac{1}{2} ({}_t u_{i,j} + {}_t u_{j,i}) \quad (3.9b)$$

$$\text{and } {}_t n_{ij} = \frac{1}{2} {}_t u_{k,i} {}_t u_{k,j} \quad (3.9c)$$

where S_{ij} is the incremental 2nd Piola-Kirchhoff stress tensor and ${}_t e_{ij}$, ${}_t n_{ij}$ are the linear and nonlinear incremental strains which are referred to the configurations at time t .

By using Equations 3.8 and 3.9, the incremental form of Equation 3.6 can be written as

$$\int_V {}_t S_{ij} \delta {}_t e_{ij} {}^t dV + \int_V {}^t \tau_{ij} \delta {}_t n_{ij} {}^t dV = {}^{t+\Delta t} W - \int_V {}^t \tau_{ij} \delta {}_t e_{ij} {}^t dV \quad (3.10)$$

To this point no approximating assumptions have been made. Equation 3.10 is an exact statement of the equilibrium of the subsequent configuration. Approximate solutions to Equation 3.10 can be obtained by using a linear relationship between the incremental stresses and strains. Thus the incremental stresses can be expressed as

$$S_{ij} = t C_{ijrs} t \epsilon_{rs} \quad (3.11)$$

By substituting Equation 3.11 into Equation 3.10 and using the following approximations

$$S_{ij} = t C_{ijrs} t \epsilon_{rs} \quad (3.12)$$

$$\text{and } \delta_t \epsilon_{ij} = \delta_t \epsilon_{ij} \quad (3.13)$$

the linearized incremental equation can be written as

$$\int_{t_V} t C_{ijrs} t \epsilon_{rs} \delta_t \epsilon_{ij} t dV + \int_{t_V} t \tau_{ij} \delta_t n_{ij} t dV = t + \Delta t W - \\ \int_{t_V} t \tau_{ij} \delta_t \epsilon_{ij} t dV \quad (3.14)$$

Due to linearization, the obtained solution may be subject to very significant error. Thus, there is an out-of-balance virtual work in the right-hand side of Equation 3.10. In order to further reduce the out-of-balance virtual work, the solution step is repeated until the difference between the external virtual work and the internal virtual work is negligible within a certain convergence measure. Equation 3.10 solved repetitively, for $k = 1, 2, 3, \dots$, can be written as

$$\int_{t+\Delta t_V(k)} t C_{ijrs}^{(k)} \Delta t \epsilon_{rs}^{(k)} \delta_t \epsilon_{ij} t dV + \\ \int_{t+\Delta t_V(k)} t \tau_{ij}^{(k)} \delta \Delta t n_{ij}^{(k)} t dV = t + \Delta t W - \\ \int_{t+\Delta t_V(k-1)} t + \Delta t \tau_{ij}^{(k-1)} \delta t + \Delta t \epsilon_{ij}^{(k-1)} t + \Delta t dV \quad (3.15)$$

where $k = 1$ corresponds to the relation in Equation 3.14.

The relation in Equation 3.15 corresponds to a Newton-Raphson iteration. The iteration is performed by updating the components of the constitutive and stress tensors on the left-hand side, which corresponds to the use of a tangent stiffness matrix during the iteration.

So far it is assumed that the loading is deformation-independent and can be specified prior to the incremental analysis. Thus, the expression in Equation 3.4 can be evaluated using

$$\int_{\Omega_S}^{t+\Delta t} \delta u_i^S \delta u_i^S dS + \int_{\Omega_V}^{t+\Delta t} \delta u_i^V \delta u_i^V dV \quad (3.16)$$

which is only possible for the concentrated loading that does not change direction as a function of the deformations. However, if the structure is not highly distorted, it is reasonable to assume that the loading is deformation-independent. As pointed out by many investigators, if the external virtual work is deformation-dependent, there would be a non-symmetric contribution to the stiffness matrix, and this is hardly effective computationally if the load steps are small and equilibrium iterations are employed. It would become more complicated if the loading is applied in sequences. In this study, it is assumed that the loading is deformation-independent, and a more detailed discussion of forming the loading matrix for each construction sequence will be given in chapter 5.

Finite Element Discretization

To solve Equation 3.14 the finite element method is used. For this purpose the domain of integration is approximated as an assemblage of discrete finite elements with elements being interconnected at nodal points on the element boundaries. To derive the

finite element equations, only the basic formulations of the finite element method will be dealt with here.

The incremental displacements within each element are assumed to be a function of the incremental displacements at the element nodal points

$$\{u\} = [N]^{(m)} \{U\}^{(m)} \quad (3.17)$$

where $[N]^{(m)}$ is the interpolation matrix for displacement, the superscript m denotes element m and $\{U\}^{(m)}$ is the global incremental displacement components at the element nodal points.

The incremental relation between the linear components of the strain and nodal displacements in element m is expressed by

$$\{e\} = [BL]^{(m)} \{U\}^{(m)} \quad (3.18)$$

where $[BL]^{(m)}$ is the linear strain-displacement transformation matrix and $\{e\}$ is the vector of incremental strains.

Variations of strains are related to variations of displacements by the same matrix $[BL]$:

$$\{\delta e\} = [BL]^{(m)} \{\delta U\}^{(m)} \quad (3.19)$$

To define the incremental nonlinear components of the strain, a vector, $\{\xi\}$, is introduced and defined as

$$\{\xi\} = [BN]^{(m)} \{U\}^{(m)} \quad (3.20)$$

where $[BN]^{(m)}$ is the nonlinear strain-displacement transformation matrix and $\{\xi\}$ is used to define those terms in U_{ij} .

Variations of $\{\xi\}$ are expressed as

$$\{\delta\xi\} = [BN]^{(m)}\{\delta U\}^{(m)} \quad (3.21)$$

Note that Equations 3.20 and 3.21 are used to define δn_{ij} .

If the loading is deformation-independent, Equation 3.9 can be written as

$$\int_{t_V} t C_{ijrs} t e_{rs} \delta t e_{ij} t dV + \int_{t_V} t \tau_{ij} \delta t n_{ij} t dV = \\ \int_{O_V} t + \Delta t f s_i \delta u_i o dS + \int_{O_V} t + \Delta t f b_i \delta u_i o dV - \int_{t_V} t \tau_{ij} \delta t e_{ij} t dV \quad (3.22)$$

Substitution of Equations 3.17, 3.18, 3.19, 3.20 and 3.21 into Equation 3.22 and a change from tensor to matrix notation gives

$$\sum_m \int_{t_V} \{\delta U\}^{(m)T} t [BL]^{(m)T} t [C]^{(m)} t [BL]^{(m)} \{U\}^{(m)} t dV + \\ \sum_m \int_{t_V} \{\delta U\}^{(m)T} t [BN]^{(m)T} t [\tau]^{(m)} t [BN]^{(m)} \{U\}^{(m)} t dV = \\ \sum_m \int_{t_V} \{\delta U\}^{(m)T} [N]^{(m)T} t + \Delta t f s\}^{(m)} o dV + \quad (3.23) \\ \sum_m \int_{t_V} \{\delta U\}^{(m)T} [N]^{(m)T} t + \Delta t f b\}^{(m)} o dV + \\ \sum_m \int_{t_V} \{\delta U\}^{(m)T} t [BL]^{(m)T} t [\tau]^{(m)} t dV$$

Since Equation 3.17 must be satisfied for all kinematically admissible virtual nodal displacements, it leads to

$$t[K]\{U\} = t+\Delta t\{Q\} - t\{F\} \quad (3.24)$$

$$\text{where } [K] = [K_L] + [K_N] \quad (3.25)$$

$$[K_L] = \sum_m \int_{t_V} \{[BL]^T t[C] [BL]\} t dV \quad (3.26)$$

$$[K_N] = \sum_m \int_{t_V} \{[BN]^T t[\tau] [BN]\} t dV \quad (3.27)$$

$$t+\Delta t\{Q\} = \sum_m \int_{o_S} [N]^{t+\Delta t\{f_S\}} 0 dS + \sum_m \int_{o_V} [N]^{t+\Delta t\{f_V\}} 0 dV \quad (3.28)$$

$$t\{F\} = \sum_m \int_{t_V} \{[BL]^T t[\tau]\} t dV \quad (3.29)$$

The required matrices for the updated Lagrangian formulation can be found in Bathe [4].

If the Newton-Raphson method is used, Equation 3.24 can be written as

$$t[K]^{(i)} \{dU\}^{(i)} = t+\Delta t\{Q\} - t+\Delta t\{F\}^{(i-1)} \quad (3.30)$$

where $\{dU\}^{(i)}$ is the vector of increments in the nodal point displacement in iteration i and

$$t+\Delta t\{U\}^{(i)} = t+\Delta t\{U\}^{(i-1)} + \{dU\}^{(i)} \quad (3.31)$$

Updated Lagrangian Jaumann Stress Rate Formulation

In geometric nonlinear analysis, before performing a new iteration within an increment, the Cauchy stresses in the updated configuration must be determined. The Cauchy stresses in the updated configuration cannot be obtained by simply adding to the Cauchy stresses in the previous configuration a stress increment which is due only to the straining of the material; that is, the calculation of the Cauchy stresses in the updated configuration must also take into account the rigid body rotation of the material, because the components of the Cauchy stress tensor also change when the material is only subjected to a rigid body rotation. Use is made of the Jaumann's stress-rate tensor, τ^J_{ij} , which is defined as

$${}^t\tau^J_{ij} = {}^t\tau_{ij} - {}^t\tau_{ip} {}^t\Omega_{pj} - {}^t\tau_{jp} {}^t\Omega_{pi} \quad (3.32)$$

where ${}^t\tau_{ij}$ is a Cartesian component of the time derivative of the Cauchy stress tensor evaluated at time t , and ${}^t\Omega_{ij}$ is a Cartesian component of the spin tensor,

$${}^t\Omega_{ij} = \frac{1}{2} \left(\frac{\partial {}^tu_i}{\partial {}^tu_j} - \frac{\partial {}^tu_j}{\partial {}^tu_i} \right) \quad (3.33)$$

Physically, the spin tensor represents the angular velocity of the material.

To obtain a physical understanding of the above tensors in the time-independent incremental analysis, consider the motion from t to time $t+\Delta t$. If the increment is infinitesimally small, then

$$\int \frac{\partial {}^tu_i}{\partial {}^tx_i} dt = \frac{\partial {}^tu_i}{\partial {}^tx_i} \quad (3.34)$$

From Equations 3.32 and 3.33 and relations in Equation 3.34, it can be shown that the Cauchy stresses at subsequent state in a 2-D formulation can be expressed as

$$t+\Delta t \tau_{11} = t\tau_{11} + \tau^J_{11} + 2t\tau_{12}w$$

$$t+\Delta t \tau_{12} = t\tau_{12} + \tau^J_{12} + (t\tau_{11} - t\tau_{22})w$$

$$t+\Delta t \tau_{22} = t\tau_{22} + \tau^J_{22} + 2t\tau_{12}w \quad (3.35)$$

$$t+\Delta t \tau_{33} = t\tau_{33} + \tau^J_{33}$$

where

$$w = \frac{1}{2} \left(\frac{\partial u_2}{\partial t x_1} - \frac{\partial u_1}{\partial t x_2} \right) \quad (3.36)$$

In deriving the governing incremental continuum mechanics equations, the principle of virtual work used is independent of the material law. For use with the rate-type incremental plasticity, Equation 3.10 can be written as

$$\int_V t C^{EP}_{ijrs} t \epsilon_{ij} \delta_t \epsilon_{rs} t dV + \int_V t \tau_{ij} \delta_t n_{ij} t dV = t + \Delta t W -$$

$$\int_V t \tau_{ij} \delta_t \epsilon_{ij} t dV \quad (3.37)$$

where $t C^{EP}_{ijrs}$ is the instantaneous constitutive relation at time t . The constitutive relation will be discussed in Chapter 4. The above equation is used to calculate the incremental displacements. The incremental stresses or Jaumann stress (rate) can be found from Equation 3.12 as

$$t\tau_{ij} = CEP_{ijrs} t\epsilon_{rs} \quad (3.38)$$

and the Cauchy stresses at the subsequent configuration at time $t+\Delta t$ can be found from the relation in Equation 3.21 as

$$t+\Delta t\tau_{ij} = t\tau_{ij} + t\tau^J_{ij} + t\tau_{ip}tw_{pj} + t\tau_{jp}tw_{pi} \quad (3.39)$$

where w_{pj} is the spin tensor in the time-independent problem with infinitesimal increments.

Limitations of Jaumann Stress-Rate Tensor

In this chapter, the updated Lagrangian formulation of the finite element method is presented. The method uses the Jaumann stress-rate tensor to update the Cauchy stresses for each iteration. While the Jaumann stress-rate tensor is usually used in the updated Lagrangian formulation, recent research shows that it is not applicable to materials exhibiting kinematic hardening.

The use of the Jaumann stress-rate tensor in the stress analysis at the finite deformation of materials exhibiting kinematic hardening incorporates rotational effects [20]. This results in oscillatory shear stress for monotonically increasing simple shear strain [60]. Taylor and Becker [90] specifically refer to the findings of Lee et al. and suggest that careful consideration must be given to the choice of stress rate for the case of kinematic or combined hardening. They, too, conclude that the Jaumann stress rate is inappropriate for this case. Corotational rates for kinematic hardening at large plastic deformations are suggested by Dafalias [20]. Lee et al. [60] proposed a modified Jaumann derivative based on the spin of the specific material directions associated with the kinematic hardening.

Kiousis et al. [55] have proposed to use a total Lagrangian formulation in which the rotational effects introduced by the Jaumann stress-rate tensor can be eliminated in the case of materials that exhibit kinematic hardening behavior. This requires that the constitutive equations used be expressed in terms of the 2nd Piola-Kirchhoff stresses. However, the available constitutive equations are defined with respect to the deforming material. In other words, they are expressed in term of the true stresses, the Cauchy stresses.

Due to the above discussion, in this study, only the isotropic models will be considered.

CHAPTER 4

CONSTITUTIVE LAWS

A constitutive law represents a mathematical model that simulates the physical stress-strain behavior of a material. The success of the application of any numerical method such as the finite element, finite difference, and boundary element methods in structural and geotechnical engineering is not only dependent upon the accuracy of the methods, but also upon how well and realistically the behavior of the material is simulated under multiaxial stresses.

The extreme difficulty in obtaining an exact solution even with the aid of digital computers is mainly due to the fact that the nonlinear stress-strain relationship in soil is far more complicated than the Hooke's law for linearly elastic materials.

Generally, the various models for defining the constitutive behavior of soils can be divided in three main groups [23]: (1) representation of given stress-strain curves using curve-fitting methods, interpolation, or mathematical functions; (2) nonlinear elasticity theories; and (3) plasticity theories. In this chapter the models that are used in this research are described.

Linear Elastic Model

For an elastic material, the state of stress is a function of the current state of deformation only. An elastic medium returns to its initial state after a cycle of loading and unloading. An elastic material in general can be nonlinear. A special case is that of linear elastic behavior. The linear elastic Hooke's law is the simplest example of a constitutive model.

A generalized Hooke's law presented by Cauchy has the form

$$\sigma_{ij} = C_{ijkl} \epsilon_{kl} \quad (4.1)$$

where σ_{ij} is the stress tensor, ϵ_{kl} is the strain tensor and C_{ijkl} is the tensor of elastic constants of the material. The specialization of Equation 4.1 for an isotropic material takes the form of [65]

$$\sigma_{ij} = \frac{E}{1+\nu} \epsilon_{ij} + \frac{\nu E}{(1+\nu)(1-2\nu)} \epsilon_{kk} \delta_{ij} \quad (4.2)$$

where E is Young's modulus, ν is Poisson's ratio and δ_{ij} is the Kronecker delta. Equation 4.2 can be expressed in matrix form

$$\{\sigma\} = [C]\{\epsilon\} \quad (4.3)$$

where

$$\{\sigma\}^T = [\sigma_{11}, \sigma_{22}, \sigma_{33}, \sigma_{12}, \sigma_{23}, \sigma_{13}] \quad (4.4)$$

$$\{\epsilon\}^T = [\epsilon_{11}, \epsilon_{22}, \epsilon_{33}, \epsilon_{12}, \epsilon_{23}, \epsilon_{13}] \quad (4.5)$$

$$[C] = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \quad (4.6)$$

In the cases of plane stress, plane strain and axisymmetric idealizations, Equation 4.3 can be simplified as

Plane Stress

$$\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{bmatrix} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \begin{bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{12} \end{bmatrix} \quad (4.7)$$

Plane Strain

$$\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{bmatrix} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \begin{bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{12} \end{bmatrix} \quad (4.8)$$

Axisymmetric

$$\begin{bmatrix} \sigma_{rr} \\ \sigma_{\theta\theta} \\ \sigma_{zz} \\ \sigma_{rz} \end{bmatrix} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 \\ \nu & 1-\nu & \nu & 0 \\ \nu & \nu & 1-\nu & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \begin{bmatrix} \epsilon_{rr} \\ \epsilon_{\theta\theta} \\ \epsilon_{zz} \\ \epsilon_{rz} \end{bmatrix} \quad (4.9)$$

Variable Moduli Model

The variable moduli model is basically an extension of the linear elastic model. The variable moduli are considered to be stress-dependent. The basic idea in the model is that the (incremental) constitutive relations are formulated directly based on the linear elastic form by simply replacing the constant moduli E and ν with variable tangential moduli E_t and ν_t . The incremental stress-strain relations are written as

$$d\sigma_{ij} = \frac{E_t}{1+\nu_t} d\varepsilon_{ij} + \frac{\nu_t E_t}{(1+\nu_t)(1-2\nu_t)} d\varepsilon_{kk} \delta_{ij} \quad (4.10)$$

Different versions of the variable moduli models have been proposed. The one by Duncan et al. [34, 35] is used in this study.

Following the suggestion of Kondner [56], the nonlinear stress-strain behavior for both clay and sand may be approximated by a hyperbolic curve (Figure 4-1). By using the hyperbolic relationship and the Mohr-Coloumb failure criterion, the tangent Young's modulus is given as

$$E_t = \left[1 - \frac{R_f(1-\sin\theta)(\sigma_1-\sigma_3)}{2c \cos\theta + 2\sigma_3 \sin\theta} \right] K p_a \left(\frac{\sigma_3}{p_a} \right)^n \quad (4.11)$$

where

R_f = failure ratio

σ_1 = major principal stress

σ_3 = minor principal stress

θ = internal angle of friction

c = cohesion

K = dimensionless constant

n = dimensionless constant

p_a = atmospheric pressure

Values of the five parameters, R_f , θ , c , K and n , in Equation 4.11 can be readily obtained from conventional triaxial tests.

A similar relation based on the hyperbolic concept was proposed by Kulhawy et al. [57] for tangent Poisson's ratio. However, as an alternative to E_t and ν_t , Duncan et

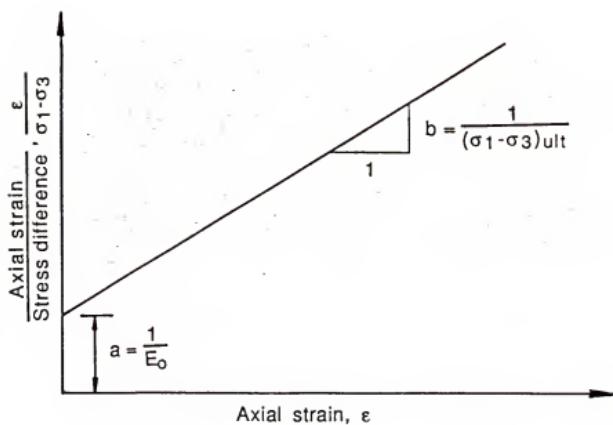
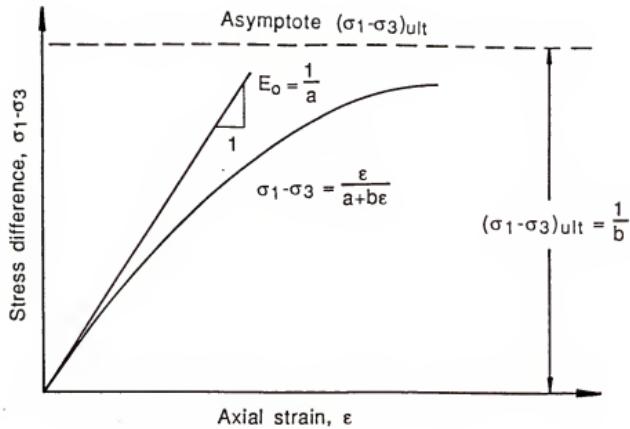


Figure 4-1 Hyperbolic representation of a stress-strain curve; (a) Hyperbolic stress-strain curve; (b) Transformed stress-strain curve

al. [34] have proposed that hyperbolic models expressed in terms of E_t and tangent bulk modulus K_t can be more conveniently utilized for a better representation of soil behavior near and after failure. Equation 4.10 can easily be expressed in terms of E_t and K_t by substituting for v_t using the relation

$$v_t = \frac{3K_t - E_t}{6K_t} \quad (4.12)$$

The expression of K_t is given as a function of the confining pressure:

$$K_t = k_b p_a \left(\frac{\sigma_3}{p_a} \right)^m \quad (4.13)$$

in which both k_b and m are dimensionless material constants, and p_a is the atmospheric pressure. Values of parameters k_b and m may also be obtained from the conventional triaxial tests. Representative values of these constants for a variety of soils may be found in Duncan et al. [34].

Plasticity Models

Most geologic media do not exhibit linear elastic behavior. Although nonlinear elastic constitutive relations have been applied to soils successfully, the main physical feature leading to a soil's nonlinear behavior is irrecoverability of strain. The theory of plasticity deals with such phenomenon and is a broad subject. Here a brief review of some of the basic theories in plasticity is given, and then three plasticity models which are used in this study are explained.

Theory of Plasticity

There are two major theories in plasticity, the deformation theory and the incremental or flow theory [50, 94]. The deformation theory is based on the assumption that there are unique constitutive relations between the total strain and current state of stress and these relations are independent of the stress path. In the flow theory, relations are expressed in terms of stress and strain increments. The incremental stress-strain relations are affected by the stress path. Since the behavior of nonlinear materials is generally dependent on the stress path, the flow theory has been widely used.

Three properties in addition to the elastic stress-strain relations are needed to describe the incremental stress-strain relationship of an elastic-plastic material based on the flow theory of plasticity:

1. a yield condition, which specifies the state of stress corresponding to the start of plastic behavior;
2. a flow rule, which relates the irrecoverable plastic strain increment to the stress state in the material;
3. a hardening rule, which defines how the yield condition is modified during plastic deformation.

These properties are adequately explained in the literature on the subject [10, 11, 13, 32, 50, 67, 97]. However, a brief description is given.

Yield Surface

If uniaxial behavior of a material is considered, the yield stress is defined as a stress point (yield point) below which the deformations are elastic and fully recoverable. For multiaxial behavior, the state of stress at a point is represented by the nine components of the stress tensor σ_{ij} , and the yield stress becomes a hypersurface in a nine dimensional space of stress.

Considering isotropic hardening, the yield condition (surface) can be written as

$$F(\sigma_{ij}, k) = 0 \quad (4.14)$$

where k is a state variable which depends on the plastic strain ϵP_{ij} . This yield condition can be visualized as a surface in 9-dimensional space of stress with the position of the surface dependent on the instantaneous value of the parameter k . The stress state where $F = 0$ implies a plastic state of stress, $F < 0$ implies an elastic state and $F > 0$ has no meaning.

On physical grounds, one can notice that the yield condition should be independent of the orientation of the coordinate system employed. In general, assuming isotropy, the yield surface is expressed in terms of the three stress invariants or the deviatoric stress invariants.

Flow Rule

When the plastic deformation occurs, the material flows under continuing application of stress. The flow rule is analogous to the relation between the flow of fluids and a potential function where the normals at any point are proportional to the velocity components at that point. When materials are stressed beyond the yield surface, the material flows with incremental plastic strains normal to a potential function at a point. Thus, a potential function G is required to calculate the plastic strain increments

$$d\epsilon P_{ij} = r \frac{\partial G}{\partial \sigma_{ij}} \quad (4.15)$$

where r is a scalar factor which controls the magnitude of the generated plastic strain increment and, like the yield function F , G is a scalar function of stress. When $F = G$, the

flow rule is called associated, and if $F \neq G$, it is termed non-associated. The potential function is found experimentally for a material.

Hardening Rule

The purpose of postulating a hardening rule is to reflect as closely as possible the history of the deformation. The yield surface may grow or undergo a rigid body motion during the plastic deformation. This phenomenon is due to the hardening. When the yield surface expands uniformly in all directions without any translation, it is called isotropic hardening. If the yield surface translates without any change in its shape and size, it is called kinematic hardening. The combination of isotropic hardening and kinematic hardening is called anisotropic hardening. In this study, the isotropic hardening is considered.

During plastic deformation, an increment of stress may cause a differential change in position of the yield surface, which can be expressed for isotropic hardening in the form:

$$dF = \frac{\partial F}{\partial \sigma_{ij}} d\sigma_{ij} + \frac{\partial F}{\partial k} dk = 0 \quad (4.16)$$

This expression is commonly known as the consistency condition. Since the hardening parameter k for isotropic hardening is a function of plastic strain, the above equation can be written as

$$dF = \frac{\partial F}{\partial \sigma_{ij}} d\sigma_{ij} + \frac{\partial F}{\partial k} \frac{\partial k}{\partial \epsilon_{P_{ij}}} d\epsilon_{P_{ij}} = 0 \quad (4.17)$$

Incremental Stress-Strain Relations

In this section, incremental relations between stresses and strains for the isotropic hardening material are derived. The total strain increments have been assumed to be the sum of the elastic and plastic strain increments. That is

$$d\varepsilon_{ij} = d\varepsilon^e_{ij} + d\varepsilon^p_{ij} \quad (4.18)$$

$$\text{or} \quad d\varepsilon^e_{ij} = d\varepsilon_{ij} - d\varepsilon^p_{ij} \quad (4.19)$$

Using the generalized Hooke's law, the stress increments can be computed as

$$d\sigma_{ij} = C_{ijkl} (d\varepsilon_{kl} - d\varepsilon^p_{kl}) \quad (4.20)$$

where C_{ijkl} is the elastic constitutive tensor.

Substitution of Equation 4.15 in Equation 4.20 gives

$$d\sigma_{ij} = C_{ijkl} (d\varepsilon_{kl} - r \frac{\partial G}{\partial \sigma_{kl}}) \quad (4.21)$$

Substitution of Equations 4.21, 4.15 in Equation 4.17 leads to

$$\frac{\partial F}{\partial \sigma_{ij}} C_{ijkl} (d\varepsilon_{kl} - r \frac{\partial F}{\partial \sigma_{kl}}) + \frac{\partial F}{\partial k} \frac{\partial k}{\partial \varepsilon^p_{ij}} r \frac{\partial G}{\partial \sigma_{ij}} = 0 \quad (4.22)$$

Solution of Equation 4.22 for r gives

$$r = \frac{\frac{\partial F}{\partial \sigma_{ij}} C_{ijkl} d\varepsilon_{kl}}{\frac{\partial f}{\partial \sigma_{mn}} C_{mn} u v \frac{\partial F}{\partial \sigma_{uv}} - \frac{\partial F}{\partial k} \frac{\partial k}{\partial \varepsilon^p_{mn}} \frac{\partial G}{\partial \sigma_{mn}}} \quad (4.23)$$

By using Equations 4.21 and 4.23, the incremental relations between stress and strain can be written as

$$d\sigma_{ij} = C^e p_{ijrs} d\varepsilon_{rs} \quad (4.24)$$

where

$$C^e p_{ijrs} = C_{ijrs} - \frac{C_{ijkl} \frac{\partial F}{\partial \sigma_{kl}} \frac{\partial G}{\partial \sigma_{mn}} C_{mnr}}{\frac{\partial F}{\partial \sigma_{mn}} C_{mnuv} \frac{\partial F}{\partial \sigma_{uv}} - \frac{\partial F}{\partial k} \frac{\partial k}{\partial \varepsilon p_{mn}} \frac{\partial G}{\partial \sigma_{mn}}} \quad (4.25)$$

If the flow rule is associated, the above equation can be written as

$$C^e p_{ijrs} = C_{ijrs} - \frac{C_{ijkl} \frac{\partial F}{\partial \sigma_{kl}} \frac{\partial F}{\partial \sigma_{mn}} C_{mnr}}{\frac{\partial F}{\partial \sigma_{mn}} C_{mnuv} \frac{\partial F}{\partial \sigma_{uv}} - \frac{\partial F}{\partial k} \frac{\partial k}{\partial \varepsilon p_{mn}} \frac{\partial F}{\partial \sigma_{mn}}} \quad (4.26)$$

Normally the elastic-plastic constitutive relation is expressed in matrix form in the finite element analysis. If the flow rule is associated, the constitutive matrix is symmetric. The incremental stress-strain matrices for the constitutive models used in this study are presented next.

Mohr-Coulomb Model

According to the Mohr-Coulomb criterion, the shear strength increases with increasing normal stress on the failure plane:

$$\tau = c + \sigma \tan \phi \quad (4.27)$$

where τ is the shear stress on the failure plane, c the cohesion of the material, σ the normal effective stress on the failure plane, and ϕ the angle of internal friction. This failure criterion is shown graphically in Figure 4-2.

The concept of Mohr circle can be used to express the Mohr-Coulomb criterion in term of principal stresses. Thus, the definition of the equation of the Mohr-Coulomb surface can be written as

$$F = \frac{\sigma_1 + \sigma_3}{2} \sin\phi - \frac{\sigma_1 - \sigma_3}{2} - c \cos\phi = 0 \quad (4.28)$$

where σ_1 and σ_3 are the major and minor principal stresses, respectively. As can be seen in Equation 4.28, the Mohr-Coulomb criterion ignores the effects of intermediate principal stress. Equation 4.28, when written in terms of invariants [88], becomes

$$F = \frac{1}{3} I_1 \sin\theta + \sqrt{J_2} \left(\cos\theta - \frac{\sin\theta \sin\phi}{\sqrt{3}} \right) - c \cos\phi = 0 \quad (4.29)$$

where I_1 is the first stress invariant, J_2 is the second deviatoric stress invariant and θ is the Lode angle which is defined in terms of the second and third deviatoric stress invariants as

$$\theta = - \arcsin \left(- \frac{3\sqrt{3}}{2} \frac{J_3}{J_2^{3/2}} \right) \quad (4.30)$$

and $-\pi/6 \leq \theta \leq \pi/6$

The elastic-plastic constitutive matrix is obtained from Equations 4.29, 4.30 and 4.26 by proper mathematical manipulation. This is shown in Appendix A. The two

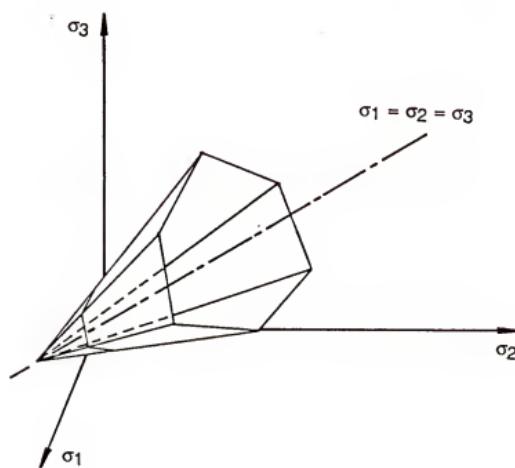
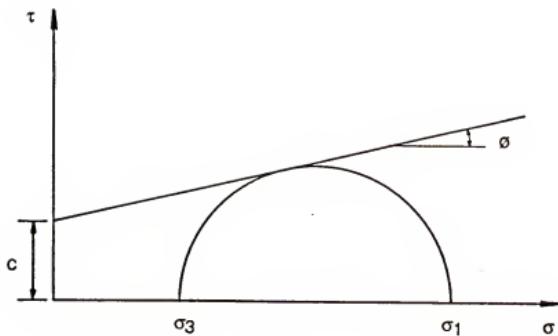


Figure 4-2 Mohr-Coulomb criterion

parameters associated with the Mohr-Coulomb failure surface, angle of internal friction ϕ and cohesion c , can be determined from conventional triaxial tests.

Drucker-Prager Model

A generalization to account for the effects of all principal stresses was suggested by Drucker and Prager [33] by using the invariants of the stress tensor. The Drucker-Prager model consists of a single failure surface at which plastic flow occurs. The function for the failure surface is

$$F = \sqrt{J_2} + aI_1 - k = 0 \quad (4.31)$$

where I_1 = first invariant of the stress tensor

J_2 = second invariant of the deviatoric stress tensor

a, k = material constants

For the plane strain condition, parameters a and k can be expressed in terms of the angle of internal friction ϕ and the cohesion c [11, 12, 23]:

$$a = \frac{\tan\phi}{\sqrt{9 + 12\tan^2\phi}} \quad (4.31a)$$

$$k = \frac{3c}{\sqrt{9 + 12\tan^2\phi}} \quad (4.31b)$$

Figure 4-3 shows the failure surface in terms of the stress invariants I_1 and J_2 . Figure 4-4 illustrates the failure surface in principal stress space. This model has been found to predict a larger volumetric strain than is observed in the laboratory.

For Drucker-Prager plasticity the constitutive matrix can be derived explicitly [23, 28]. Only the expression for the plane strain condition is given here.

$$\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \\ \sigma_{33} \end{bmatrix} = 2G \begin{bmatrix} 1-T_1\sigma_{11}-R_1 & -(T_1\sigma_{22}+R_1) & -(T_1\sigma_{12}) & -(T_1\sigma_{33}+R_1) \\ -(T_2\sigma_{11}+R_2) & 1-T_2\sigma_{22}-R_2 & -(T_2\sigma_{12}) & -(T_2\sigma_{33}+R_2) \\ -T_1\sigma_{12} & -T_2\sigma_{12} & 1/2-C\sigma_{12}^2 & -(T_3\sigma_{12}) \\ -(T_2\sigma_{11}+R_3) & -(T_3\sigma_{22}+R_3) & -(T_3\sigma_{12}) & 1-T_3\sigma_{33}-R_3 \end{bmatrix} \begin{bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{12} \\ \epsilon_{33} \end{bmatrix} \quad (4.32)$$

where $T_n = A + C\sigma_{nn}$ and $R_n = A\sigma_{nn} + B$; $n = 1, 2, 3$. Here a repeated subscript does not mean summation. A, B and C are defined as

$$A = \frac{h}{pk} \quad (4.33)$$

$$B = \frac{2h^2}{1+9a^2\frac{K}{G}} - \frac{3\nu K}{E} \quad (4.34)$$

$$C = \frac{1}{2kp\sqrt{J_2}} \quad (4.35)$$

$$p = \frac{\sqrt{J_2}}{k} \left(1 + \frac{9a^2 k}{G} \right) \quad (4.36)$$

$$h = -\left(\frac{3aK}{2G} + \frac{1}{6\sqrt{J_2}} \right) \quad (4.37)$$

where K = bulk modulus

E = Young's modulus

ν = Poisson's ratio

G = shear modulus

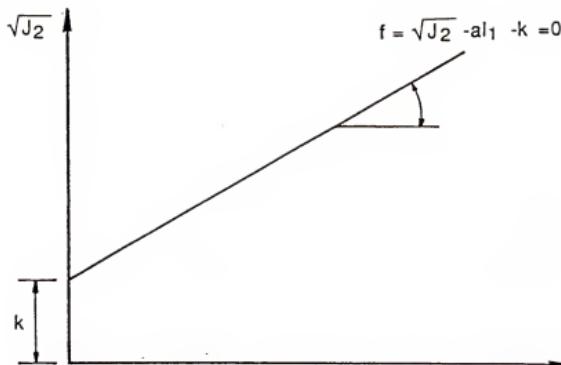


Figure 4-3 Drucker-Prager failure surface I_1

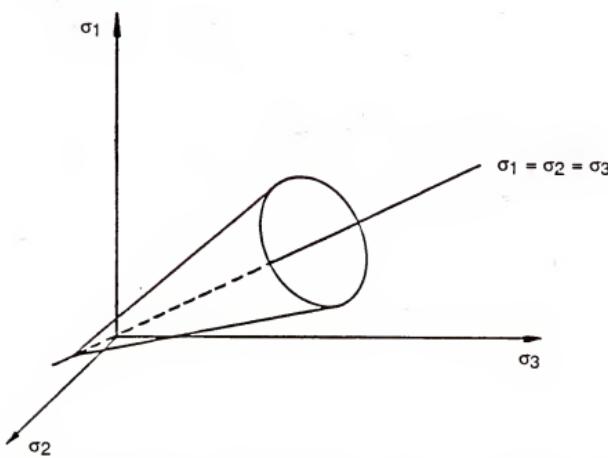


Figure 4-4 Drucker-Prager Failure surface in 3-D stress space

Cap Model

Cap models were proposed to overcome some of the deficiencies of the Drucker-Prager model. Instead of a single failure surface as for the Drucker-Prager model, the Cap model has a failure surface and an additional strain-hardening yield surface. This additional yield surface is usually referred to as the cap.

The cap changes as plastic deformation takes place. It functions to allow plastic flow for loading paths which do not approach the material's failure surface. Since most geologic materials exhibit plastic deformation long before failure, the cap model generally proves to be a more accurate model than the Drucker-Prager model. Also, the cap prevents the model from predicting excessive dilation at failure.

The model chosen for this research was developed by Sandler and Rubin [80]. The function of the failure surface [89], which is an extension of the Drucker-Prager failure surface, is given by

$$F_f(I_1, \sqrt{J_2}) = A - \theta I_1 - C e^{BI_1} \sqrt{J_2} = 0 \quad (4.38)$$

where A, B, C and θ are material parameters.

The function for the cap is chosen to be an elliptically shaped function. Sandler and Rubin give the following elliptical form

$$F_c(I_1, \sqrt{J_2}, k) = \frac{1}{R} \{ [X(k) - L(k)]^2 - [J_1 - L(k)]^2 \}^{1/2} - \sqrt{J_2} = 0 \quad (4.39)$$

in which R is the ratio of the major to the minor axis of the elliptic cap; X and L are functions of k defined below.

$$X(k) = k - R F_f(k) \quad (4.40a)$$

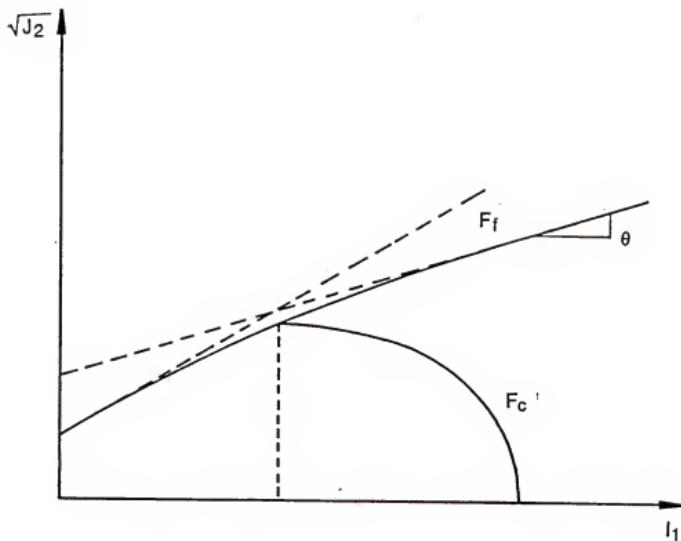


Figure 4-5 Cap model

$$L(k) = k \quad \text{if } k < 0 \quad (4.40b)$$

$$L(k) = 0 \quad \text{if } k \geq 0 \quad (4.40c)$$

The hardening parameter, k , is related to the plastic volumetric strain by the following equations.

$$k = \varepsilon P_{kk} = W \{ \exp[DX(k) - 1] \} \quad (4.41)$$

where W and D are material parameters.

The model required the evaluation of nine parameters from various laboratory tests. Two of the parameters are the linear elastic constants, Young's modulus and Poisson's ratio. For stress states not on the cap or failure surface, the material is assumed to follow the generalized Hooke's law. Figure 4-5 illustrates the Cap model.

The elastic-plastic constitutive matrix is obtained from Equations 4.38, 4.39, 4.40, 4.41 and 4.26 by proper mathematical manipulation. This is shown in Appendix II.

Interface Elements

Interface elements are used to model the interaction between dissimilar materials. A number of interface elements have been proposed and applied with some success [40, 42, 43, 48]. The interface element used for this study has been under development for the past several years [62, 81]. The basic proposals behind the interface element are as follows:

1. The same element formulation as for a solid element is used.
2. An arbitrary "small" thickness of the interface is chosen, the limiting factor being the numerical stability of the resulting element equations.
3. The interface behavior is modeled through the use of a modified constitutive

law.

4. A minimum of three independent parameters are required for the constitutive model.

The main advantages of the element are its compatibility with solid elements, and its simplicity compared to other types of interface elements. The constitutive matrix is the heart of the interface element. Two interface models used in this study are described below.

Linear Elastic Interface Model

The linear elastic model under plane strain conditions is defined as

$$[C] = \begin{bmatrix} \frac{1-\nu^2}{E} & \frac{-\nu(1+\nu)}{E} & 0 & \frac{-\nu(1+\nu)}{E} \\ \frac{-\nu(1+\nu)}{E} & \frac{1-\nu^2}{E} & 0 & \frac{-\nu(1+\nu)}{E} \\ 0 & 0 & \frac{1}{G} & 0 \\ \frac{-\nu(1+\nu)}{E} & \frac{-\nu(1+\nu)}{E} & 0 & \frac{1-\nu^2}{E} \end{bmatrix} \quad (4.42)$$

where E = Young's modulus

ν = Poisson's ratio

G = shear modulus

The two parameters E and ν are generally taken to be the same as for one of the solid elements it adjoins. The shear modulus G can be determined from direct shear tests.

Equation 4.42 indicates that the shear behavior is independent of the normal stresses. Since most observed interface shear behavior is highly dependent upon the normal stress, this model is limited. The following model considers the shear modulus is a function of the normal stress.

Hyperbolic Interface Model

A model which allows better prediction of the interface behavior is a modification of the hyperbolic model used for solid elements. The shear modulus [17, 18] is given by

$$G = \left(K \gamma_w \left(\frac{\sigma_n}{p_a} \right)^n \left(1 - \frac{R_f \sigma_{xy}}{c + \sigma_n \tan \phi} \right)^2 \right) \quad \text{if } \sigma_n \text{ compressive} \quad (4.43a)$$

$$G = 0 \quad \text{otherwise} \quad (4.43b)$$

where K = proportionality constant

γ_w = unit weight of water

σ_n = stress normal to the interface

p_a = atmospheric pressure

R_f = failure ratio

σ_{xy} = shear stress

c = adhesion of the interface

ϕ = friction angle of the interface

n = dimensionless constant

All of the parameters may be obtained from direct shear tests.

CHAPTER 5

SIMULATION OF CONSTRUCTION SEQUENCES

In this chapter, the construction sequences to be dealt with in this study are examined and the loading vector corresponding to the simulation of each sequence in the finite element solution is presented. The loading vectors can be formulated for bodies subjected to small as well as large deformations. The formulation of loading vector, if not specified, is in updated coordinates for problems with large strain and large deformations.

Insitu Stresses

Before the construction starts, the mass of the region is subjected to insitu stresses. A knowledge of the insitu stresses is a prerequisite in simulating construction sequences. Most natural deposits have undergone a complex stress history of loading, unloading or even chemical reaction which makes it extremely difficult to predict the insitu stresses. To measure the complete insitu stresses may often turn out to be a difficult, confusing, and expensive task.

One of the most common methods for predicting the insitu stresses is that the lateral (horizontal) effective stress depending on the geological history is related to the effective vertical stress created by the gravity of the overburden through a factor K_0 called the coefficient of lateral earth pressure [8]. The values of K_0 can be obtained from field or laboratory experiments. Mayne and Kulhawy [66] proposed a simple empirical formulation to predict approximate values of K_0 after reviewing laboratory data from over 170 different soils.

The method chosen here was to do a cycle of finite element analysis to compute the insitu stresses from applied loads due to the unit weight of the region. The applied load can be computed as

$$\{Q\} = \sum_m \int_V [N]^{(m)T} \{\gamma\}^{(m)} dV \quad (5.1)$$

where $[N]$ is the displacement interpolation matrix and $\{\gamma\}^{(m)}$ is the unit weight of soil for element m . In the case with water in soil, the Buoyant unit weight should be used. The proper lateral stresses, which are equal to K_0 times effective vertical stresses, are obtained by choosing the value of Poisson's ratio according to the following equation:

$$\nu = K_0 / (1+K_0) \quad (5.2)$$

Because the equation is based on linear elasticity, the computation of the insitu stresses is performed with a linear constitutive law for all materials. All displacements are set to zero after obtaining the insitu stresses since the desired quantities are the stresses.

Dewatering

Dewatering is modeled in a rather simple manner. It is assumed that the groundwater levels before and after pumping are horizontal. In this case, the change in the state of stress caused by dewatering can be included in the finite element analysis as an equivalent load that modifies the unit weight in the soil. Such an equivalent load can be computed as

$$\{Q\} = \sum_m \int_V [N]^{(m)T} \{\gamma_w\}^{(m)} dv \quad (5.3)$$

where $\{\gamma_W\}^{(m)}$ is the added apparent unit weight for element m in the dewatered region. The added weight occurs due to the loss of the Buoyant force of water on the region. For nonlinear problem, the dewatered region may be subdivided into more horizontal layers and solutions for each dewatering step found.

If dewatering is done in slow draining media such as clay soils, the groundwater level between wells will not be horizontal. A method which incorporates consolidation would be desirable for accurate results [73].

Excavation

Most available procedures for simulating excavation are based on the procedure originally proposed by Goodman and Brown [41]. In this procedure, the insitu stresses are first determined. Excavation is simulated in a number of increments. As shown in Figure 5-1, the excavation surface is considered to be a stress-free surface. This condition is satisfied by applying equivalent nodal forces which are equal but opposite in direction to the existing stresses along the surface at a given increment of excavation. The displacements and stresses are then calculated and added to the values of the previous step.

Two methods have been used to formulate the equivalent nodal forces. The first one involves using the stresses computed at the nodal points directly in computing the equivalent nodal forces. The stresses are usually evaluated at internal points inside an element first, and then, through an interpolation function, they are extrapolated to the nodal points. After the stresses at the nodal points are established, the equivalent nodal forces are obtained from equilibrium considerations.

It has long been recognized that there are difficulties in evaluating accurately the stresses along the excavation surface by using the above method. Christian and Wong [14] pointed out that excavation simulation in linearly elastic material is independent of the number of increments used for the excavation process, but for the finite elements

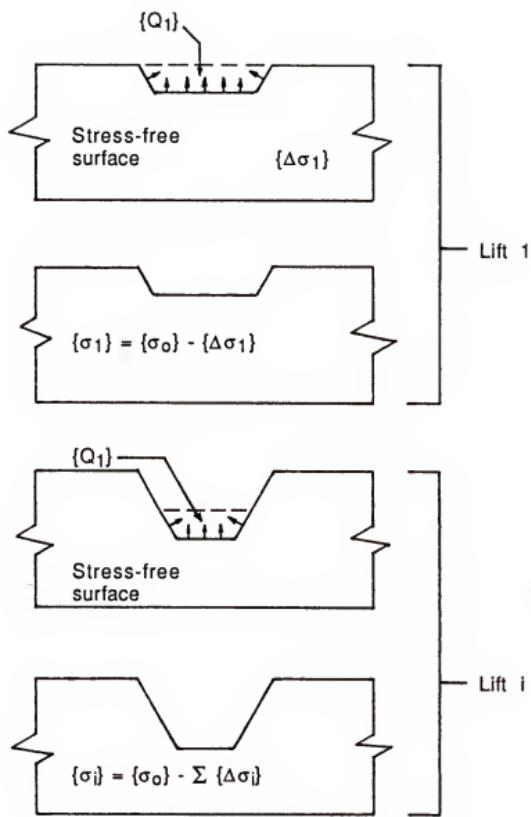


Figure 5-1 Simulation of excavation

they used, this criterion did not hold good. In other words, the problem can be dependent on the number of excavation increments. This error can become worse when one tries to simulate excavation in nonlinear materials because nonlinear problems are usually solved in a number of increments and the errors may be cumulative. Christian and Wong indicated that the errors may be caused by the factor that the lower-order linear triangular and four-noded quadrilateral elements are inadequate to model the stress gradients near the corner of the excavation. Although quadratic isoparametric elements can improve the results compared with using the lower-order elements in the same mesh, Huang [52] noted that gradients at the corner of an excavation are so steep that a relatively coarse mesh does not model them well.

The second method makes use of overall equilibrium to compute the equivalent nodal forces. In this method, the stresses are usually evaluated at Gauss points inside an element first, and then, instead of computing the nodal stresses, the equivalent nodal forces due to the stresses at Gauss points are calculated. An evaluation of the equivalent nodal forces can be accomplished by considering the equilibrium of the excavated body.

Mana [64] used eight-noded isoparametric elements in simulation of excavation, where the above method was utilized for computing the equivalent nodal forces for creating the stress-free surface. He simulated excavation in linear elastic material by using one step and then three steps, showing that the obtained results from both simulations were identical. He analyzed two cases of braced excavation constructed in soft clays, and compared the prediction with field observations.

The second method is used in this study and is extended to include the effect of large deformations. A more detailed description of this method is given below.

As shown in Figure 5-2, the excavated area is treated as a free body. If the free body is in equilibrium, reactions are required to add at nodal points on the excavated surface to counteract the acting forces. The reactions, which are equal to the equivalent nodal forces due to excavation, can be found by forming equilibrium residuals of the free

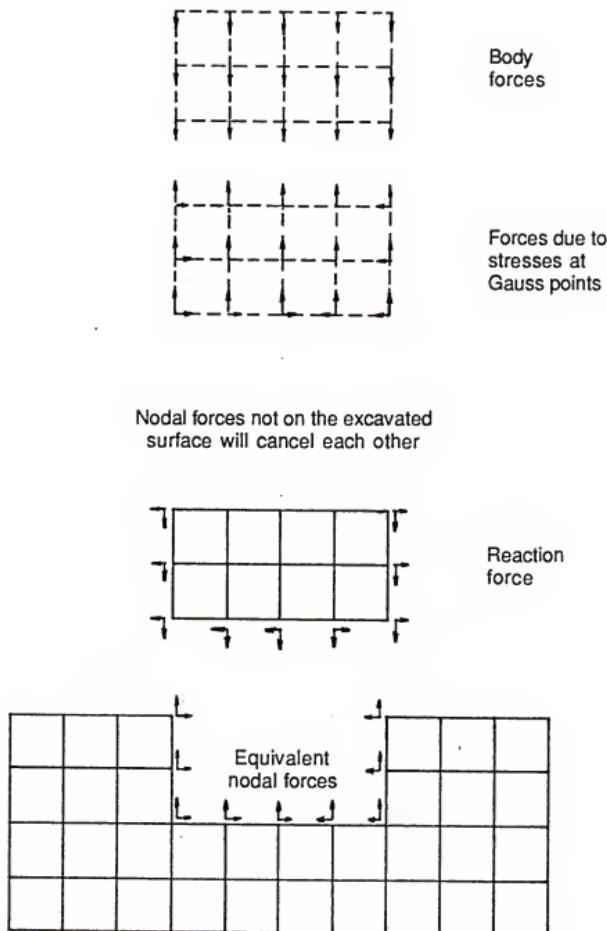


Figure 5-2 Equivalent nodal forces

body. In the finite element analysis, the procedure to obtain equilibrium residuals proceeds as follows:

- (1) Compute the equivalent nodal loads due to the stresses at the integration points in the excavated body by

$$\{F\} = \sum_m \int_V [B]^{(m)T} \{\sigma\}^{(m)} dV \quad (5.4)$$

where $[B]^{(m)}$ is the strain-displacement transformation matrix and $\{\sigma\}^{(m)}$ is the equivalent stress vector at Gauss points for element m.

- (2) Compute the equivalent nodal loads due to the unit weight of the excavated body by

$$\{Q\} = \sum_m \int_V [N]^{(m)T} T\{\gamma\}^{(m)} dV \quad (5.5)$$

For problems with large strains and large displacements, the above vector is still formulated using original coordinates because the equivalent nodal loads due to the unit weight of the region are formulated using original coordinates.

- (3) Compute the equivalent nodal loading vector, $\{E\}$, due to the effect of excavation by

$$\{E\} = \{F\} - \{Q\} \quad (5.6)$$

Embankment

Embankment is the process of adding material to the domain. In contrast to excavation simulation, modeling of embankments is fairly simple and does not contain many of the problems associated with excavation.

Embankment is simulated in a number of sequences. In each sequence, the equivalent nodal forces due to the added material are computed by

$$\{F\} = \sum_m \int_V [N]^{(m)T} \{\gamma\}^{(m)} dV \quad (5.7)$$

The displacements, stresses and strains are then calculated and added to the values of the previous step.

Adding and Deleting Bars

Tie backs and other reinforcement such as struts, anchors and braces are modeled using one-dimensional bar elements, that transmit only axial loads. The simulation of adding bars to the domain proceeds as follows:

- (1) If the bar is prestressed, place equal but opposite forces at the nodal points where the bar is attached to the solid elements.
- (2) Compute the effect due to the prestressing forces. Increment the forces for nonlinear problem.
- (3) Add the bar element stiffnesses to the global stiffness matrix to account for the presence of the reinforcement in future sequences.

The simulation of deleting bars from the domain proceeds as follows:

- (1) Apply forces which are equal but opposite in the direction to bar stresses at the nodal points where the bar is attached to the solid elements.

- (2) Compute the effect due to the applied nodal forces. Increment the forces for nonlinear problems.

Reestablishment of the Water Table

Raising the water table is the process of adding water to the domain. It is assumed, as in dewatering, that the groundwater levels before and after the rise of the water table are horizontal. The simulation of adding water to the domain proceeds as follows:

- (1) Use Equation 5.3 to compute the equivalent nodal forces due to the removed apparent weight of the region in which water is added. The removed weight occurs due to the presence of the Buoyant force on the region.
- (2) Compute the displacements and stresses due to the forces calculated above.

Concentrated Loading

The simulation of adding surface loads to the domain is achieved by directly inputting the concentrated forces to the specified nodal points. If the loading is uniform, its equivalent nodal forces need to be computed first.

CHAPTER 6

COMPUTER PROGRAM

The nature of the nonlinear finite element program with construction sequences offers several challenging problems in deriving the mathematical formulations and translating them into a practical computer program. For example, one of them is the use of the tangent stiffness method in the nonlinear finite element equations. The other problem is to consider loading and unloading in the program when dealing with construction sequences. These problems will be discussed after a brief description of the computer program.

FEMCON -- An Overview

Based on the present technology of the digital computers, it is reasonable to state that many structural problems, whether linear or nonlinear, can be solved by use of the finite element method, with the only limitations being the cost of computations and proper modeling of the nonlinear material behavior. To some extent, the computational cost of a finite element analysis may be reduced by more efficient solution techniques and effective utilization of in-core and out-of-core storage for handling large size problems. It is anticipated that future research will result in the development of specialized elements for a particular analysis, improved solution techniques for solving nonlinear problems, and new constitutive models representing the nonlinear behavior of materials. Therefore, a finite element program may become out-dated unless it is written in a manner that can be easily modified and updated.

FEMCON is written in FORTRAN 77 and should be easily adaptable to various computers. The program is constructed in a modular form so that modifications may be

easily implemented. Although it is originally developed to study nonlinear soil-structure interaction problems with or without construction sequences, it can be used or modified to solve a wide variety of solid mechanics problems. The nonlinearities may be of both geometric and material origin. The program implements the following programming techniques usually found in general codes:

- (1) Dynamic storage allocation
- (2) In-core and out-of-core data storage
- (3) In-core and out-of-core skyline equation solvers

A more detailed description of Program FEMCON can be found in its user's guide written by the author [95]. Appendix C contains an input guide for the program.

Overall Program Logic

FEMCON consists of a main program and three types of functional blocks. The main program controls the logical flow of information through the various functional blocks. A functional block is executed by reading a data record containing the name of the block in the main program. To each functional block there is a control subroutine named BLnnnn and an execution subroutine named EXnnnn. Figure 6-1 shows the key operations of the program. Subroutine BLnnnn executes the following preliminary operations of block 'nnnn':

- (a) Reads various control parameters required for the execution of the program;
- (b) Allocates memory to various arrays using dynamic memory storage techniques;
- (c) Calls subroutine EXnnnn.

Subroutine EXnnnn executes all the operations of block 'nnnn'. Its execution may require that some utility subroutines be called.

Functional blocks in FEMCON can be divided into basic functional blocks, loading functional blocks and execution functional blocks. The basic blocks are used for the

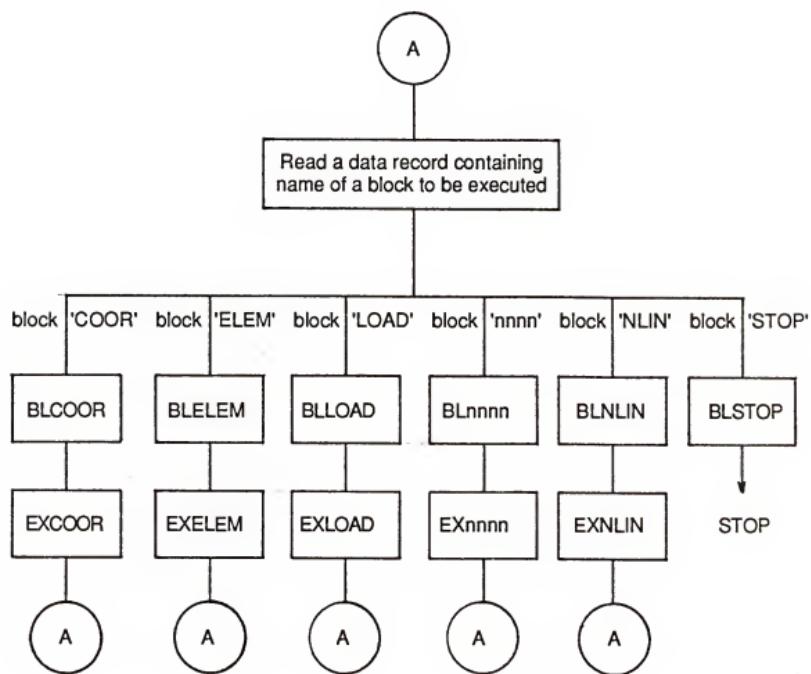


Figure 6-1 Key operations in FEMCON

input, verification and organization of the data required to define a problem. The loading blocks use the data base constructed by basic blocks to generate the loading data required in a system of equations of a problem. The execution functional blocks assemble and solve a system of equations of a problem using the data base constructed by basic and loading functional blocks. The solution to a problem can be obtained by properly managing the functional blocks. The flow chart for FEMCON is shown in Figure 6-2. To solve a problem, the user needs to do the following steps:

Step 1: Use the basic functional blocks in the following order:

- Block 'IMAG' (echo input data, optional)
- Block 'COOR' (coordinates)
- Block 'COND' (boundary conditions)
- Block 'PREL' (material properties)
- Block 'ELEM' (element data)

Step 2: Choose one or more of the following loading blocks:

- Block 'CONC' (concentrated loads, no construction sequence)
- Block 'BODY' (body forces, no construction sequence)
- Block 'INSI' (body forces, construction sequence)
- Block 'DWAT' (equivalent nodal forces due to dewatering)
- Block 'EXCA' (equivalent nodal forces due to excavation)
- Block 'EMBA' (equivalent nodal forces due to backfill or concrete placement)
- Block 'BARA' (installation of bar elements)
- Block 'BARD' (equivalent nodal forces due to the removal of bar elements)
- Block 'LOAD' (concentrated loads, construction sequence)
- Block 'FWAT' (equivalent nodal forces due to the reestablishment of the new water table)

Step 3: Choose one of the following execution blocks:

Main Flow Chart

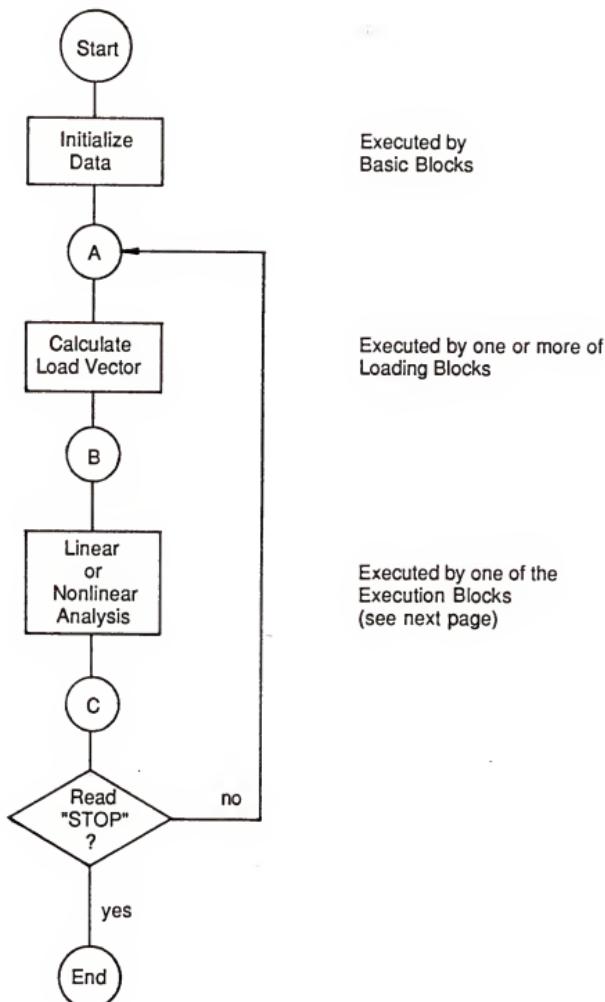
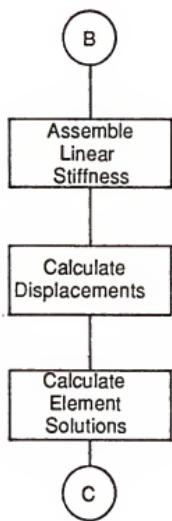


Figure 6-2 FEMCON flow chart

Linear Analysis Flow Chart



Nonlinear Analysis Flow Chart

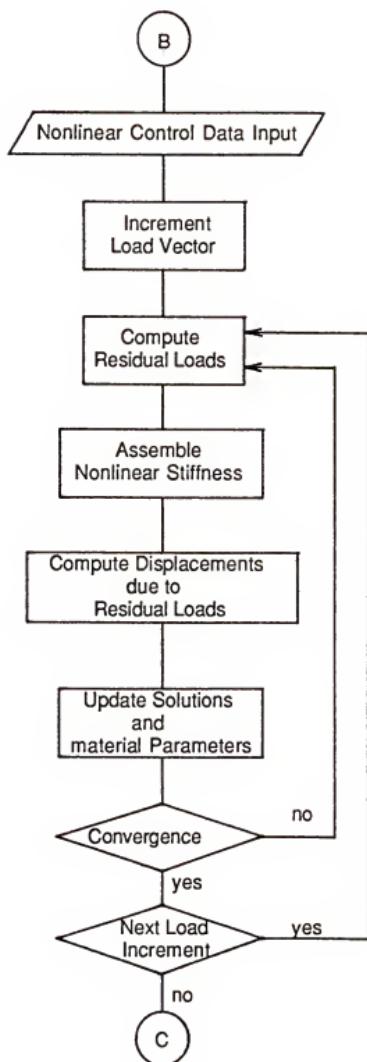


Figure 6-2 continued

- Block 'LINM' (in-core linear solution)
- Block 'NLIN' (in-core nonlinear solution)
- Block 'LIND' (out-of-core linear solution)
- Block 'NLND' (out-or-core nonlinear solution)

For problems with construction sequences, steps 2 and 3 must be repeated for each sequence. FEMCON also has some optional utility blocks:

- Block 'COMT' - writes comments in the output.
- Block.'PRNT' - prints out the solution for specified elements.

Solution of Nonlinear Equations

As described in Chapter 2, nonlinear equations can be solved using the tangent stiffness method, the modified tangent stiffness method or the constant stiffness method. But, due to the mathematical complexities in deriving the tangent stiffness matrix for the plasticity models, the constant stiffness method is usually adopted in computer programs using elastic-plastic soil models [61]. The drawback of the constant stiffness method is that a large number of iterations within each loading increment is required for a converged or accurate solution. It is observed by this author that an iteration number of 100 or even more is usually necessary to reach the same allowable tolerance of the tangent stiffness method. An attempt was made to adopt the tangent stiffness method and the tangent stiffness matrices for Mohr-Coulomb and Cap plasticity models were derived in Appendices A and B respectively. The nonlinear problems solved in this study are generally obtained using the tangent stiffness method, and the incremental solutions of them can usually converge in less than ten iterations and in some cases in three iterations. It is possible that problems that can not be solved with the tangent stiffness method exist. One of these problems is that a uniform loading is applied to the Cap model material under conditions of uniaxial strain. This is shown and explained in the next chapter.

In FEMCON, the Euclidian vector norm is used as a convergence test [30]. Convergence of the iterative solutions is measured by comparing the norm of the residual displacements to the norm of the total displacements:

$$\|\eta\| = \frac{\sqrt{\{dU_i\}\{dU_i\}}}{\sqrt{\{U_i\}\{U_i\}}}$$

The iterative process is terminated when

$$\|\eta\| < \epsilon$$

where ϵ is an arbitrarily selected small number. The nonlinear solutions were obtained in this study using a tolerance on the order of 10^{-8} .

Loading and Unloading Solution

The tangent stiffness is the preferred method to solve nonlinear equations. However, Tillerson, et al. [92] report that the tangent stiffness method has been widely used for geometrically nonlinear problems but, difficulties may arise for problems involving material nonlinearities with unloading. Before discussing how to solve nonlinear problems with unloading, it is beneficial to examine the unloading and reloading behavior.

Experimental observations of several investigators (eg., Duncan and Chang, 1970; Holubec; 1968; Makhlof and Stewart, 1965) [35, 51, 65] have indicated that the unloading and reloading behavior of many soils is nearly linear and elastic in nature, and is independent of the stress levels at which unloading starts, particularly when unloading occurs at stress levels not close to failure. For example, unloading and reloading curves at different stress levels, A and B, in Figure 6-3 have essentially the

same slope, which is nearly the same as the slope of the initial tangent in primary loading. (actual behavior of soils shows a small hysteresis loop). Based on these observations, the unloading and reloading to the maximum previous stress level may be approximated by an isotropic linear elastic model. If uniaxial behavior of a material is considered, unloading can be easily visualized from the 1-D stress-strain curve as shown in Figure 6-3. However, for multiaxial behavior, the theories of plasticity described in chapter 4 are used. Figure 6-4 shows a yield surface with a current stress state of a stress point lying on a yield surface. An incremental stress vector $\Delta \sigma_{ij}$ is applied. If the direction of $\Delta \sigma_{ij}$ is inward (path OB), then unloading occurs producing only elastic strains. If the direction of the vector is outward (path OA), both plastic and elastic deformations occur.

From the above discussion, the constant stiffness method can be used to solve nonlinear problems with unloading since it always uses the linear elastic model to formulate the stiffness matrix and unloading is a linear elastic behavior. However, when the tangent stiffness method or the modified tangent stiffness method is applied to nonlinear problems, it was observed by the author that the stress point on the yield surface after unloading may not fall within the yield surface. Thus, the linear elastic unloading behavior can not be detected and the correct stiffness can not be reassembled.

Here a new method was proposed by the author. This method takes advantage of both the constant stiffness method and the tangent stiffness method. The first iteration in each load increment uses constant stiffness method to detect the linear elastic unloading behavior and define the stress state for all Gauss (integration) points. If the stress state is found linear-elastic for a particular Gauss point then the rest of the iterations for that particular point are performed using linear stiffness matrix. On the other hand, if the stress state is on the yield surface then the subsequent iterations in that particular load increment are performed using tangent stiffness matrix. Thus the first iteration in each increment is used to check if the elastic unloading behavior occurs and thereafter

the tangent stiffness method is used to accelerate the convergence. This method has been tested extensively in this study considering both geometric and material nonlinearities. A simple unloading-reloading example is shown in the next chapter to verify this method. The two field problems in Chapter 9 were solved by this method. Also, some of the examples with monotonically increasing loading have also been tested by the tangent stiffness method as well as the proposed method in Chapter 7 and both solutions show identical results. Hereafter, the proposed method will be called the mixed stiffness method. Figure 6-5 shows how the mixed stiffness works with the incremental method in a one-dimensional problem.

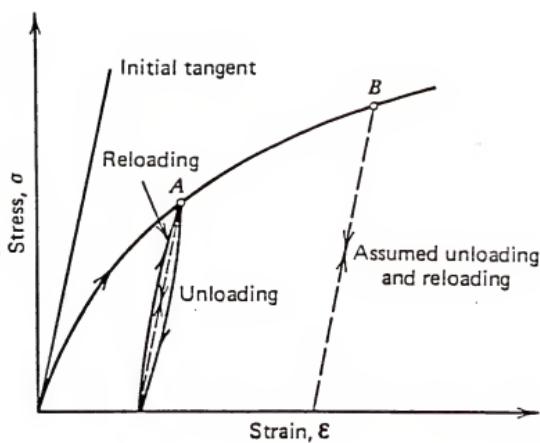


Figure 6-3 Approximate representation of loading-unloading

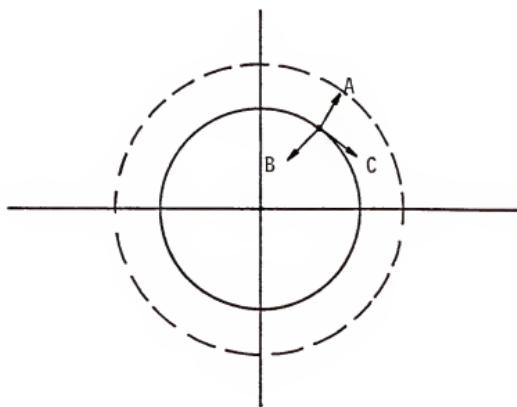


Figure 6-4 Schematic of yield surface and subsequent yield surface

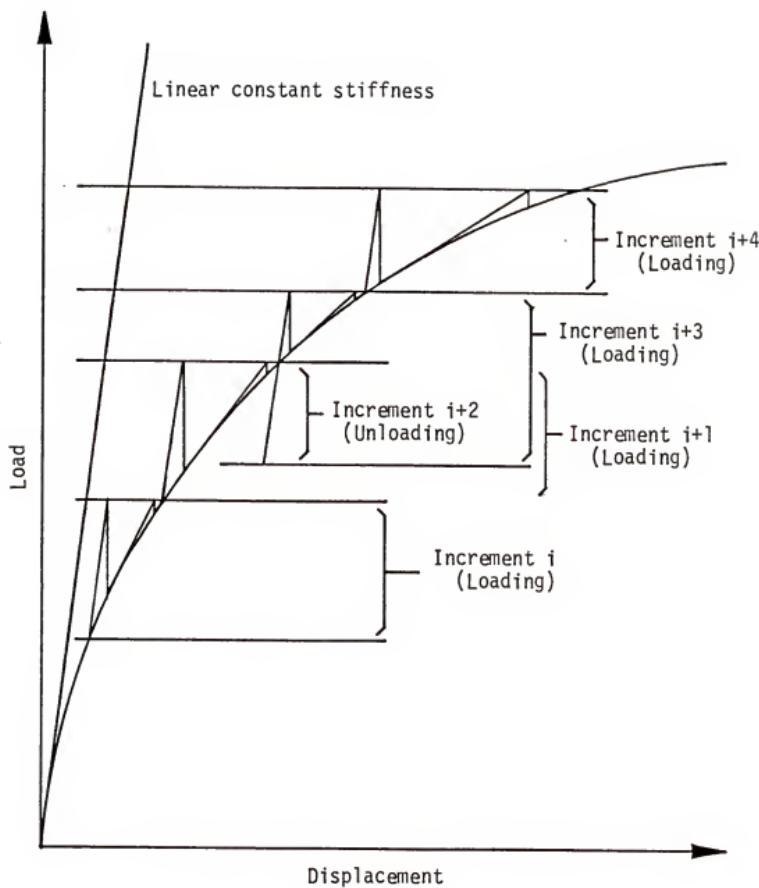


Figure 6-5 Mixed stiffness method

CHAPTER 7

SOME ANALYTICAL VERIFICATIONS

In this chapter, the computer program developed in this study is used to solve some preliminary problems. The obtained results are compared with the exact solutions or with solutions obtained by other investigators.

Plane-Strain Elastic Large Deformation Problem

Consider the plain strain problem shown in Figure 7-1a in which an elastic specimen is subjected to a uniform stress on the top side and is bounded on the other three sides by smooth rigid walls. The linear solution to the problem, which relates the stress to the small strain, can be obtained from equation 4.8 as

$$\sigma = E' \epsilon \quad (7.1)$$

where

$$E' = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \quad (7.2)$$

and $\epsilon = u / l_0$ (7.3)

u is the vertical displacement due to the applied stress and l_0 is the original height of the specimen. However, if deformations are large, the stress is related to the so-called logarithmic strain, that is,

$$\sigma = E' \epsilon_l \quad (7.4)$$

where

$$\epsilon_l = \int_0^u \frac{du}{l_o - u} = \ln\left(\frac{l_o}{l_o - u}\right) = \ln\left(\frac{1}{1 - \frac{u}{l_o}}\right) \quad (7.5)$$

Thus, the exact solution is

$$\sigma = E' \ln\left(\frac{1}{1 - \frac{u}{l_o}}\right) \quad (7.6)$$

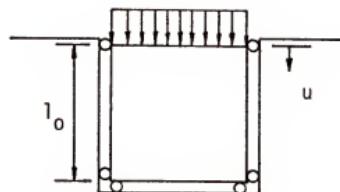
A non-dimensional plot of Equation 7.6 is shown in Figure 7-1b along with two numerical solutions, one obtained using ten increments and one with fifty increments. It appears that the approximate solution converges towards the true solution as the number of loading increments increases.

Settlement and Collapse Calculations of Footings

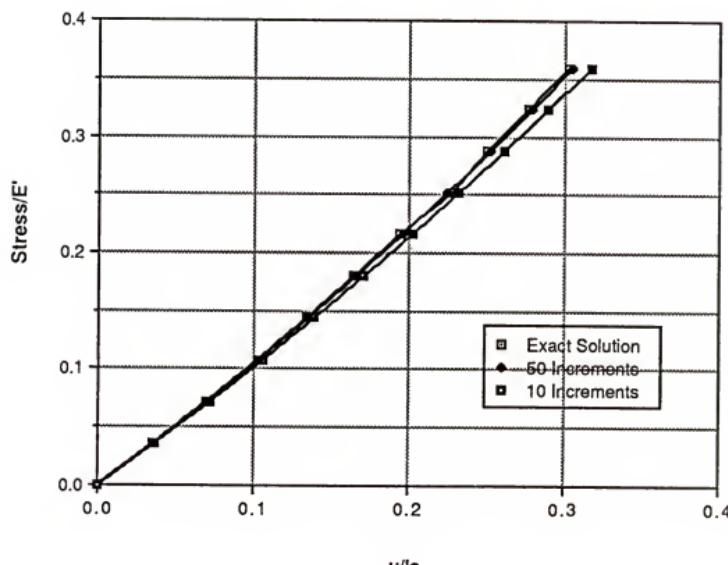
In order to verify the plastic soil models used in the computer program, a strip footing is analyzed using different soil models, and the results are compared with the solutions reported in the literature.

Strip Footing on Von Mises Material

In the first case, a flexible strip footing is supported on a homogeneous isotropic soil layer. This problem has been solved by Davidson and Chen [21] using two dimensional constant strain triangular elements for the case of small deformations. The dimensions for the problem are illustrated in Figure 7-2 and the soil properties are

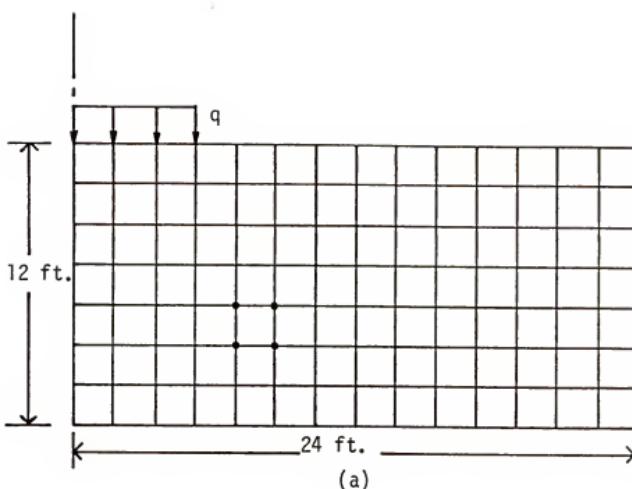


(a)

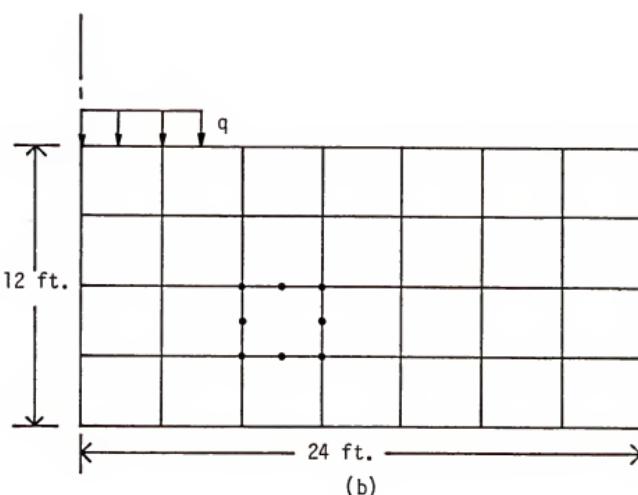


(b)

Figure 7-1 Plane-strain elastic large deformation problem; (a) Test problem;
 (b) A non-dimensional plot



(a)



(b)

Figure 7-2 Mesh geometry for footing analysis; (a) Mesh 1; (b) Mesh 2

$$E = 30,000 \text{ psi}$$

$$\nu = 0.3$$

$$c = 17.5 \text{ psi}$$

$$\phi = 0^\circ$$

The strip footing is analyzed as a plane strain problem. The Drucker-Prager model is employed in the analysis. It should be noted that for soil with an internal friction angle equal to zero, the Drucker-Prager model reduces to the Von Mises yield criterion.

The computed load-displacement curve at the center of the footing is shown in Figure 7-3. It can be seen that the results of settlement agree well with the solution by Davidson and Chen. The predictions with small and large strain assumptions appear to yield similar results. This could be due to the fact that for this problem, the deformations in the material are governed predominantly by the material nonlinearity rather than by geometric nonlinearity. Figure 7-3 also shows the effect of element selection on the response of the strip footing. The analysis using the linear isoparametric element display the "locking" phenomenon discussed by Nagtegaal et al. [71]. The analysis using the parabolic isoparametric element is consistent with the exact collapse load. It is interesting to note that the second mesh provides a better solution with fewer elements, degrees of freedom, and total gauss points. Therefore, it is expected that the use of parabolic elements will be better and more economical. All obtained solutions used the same number of loading increments and a convergence tolerance of 10^{-8} .

Strip Footing on Drucker-Prager and Cap Material

In the second case, the strip footing in Figure 7-2 was further analyzed by using the Drucker-Prager and Cap models with the following material properties.

Author's Solutions:

- Isoparametric linear element (small strain)
 - Isoparametric parabolic element (small strain)
 - ✗ Isoparametric linear element (large strain)
 - + Isoparametric parabolic element (large strain)
- △ Davidson and Chen

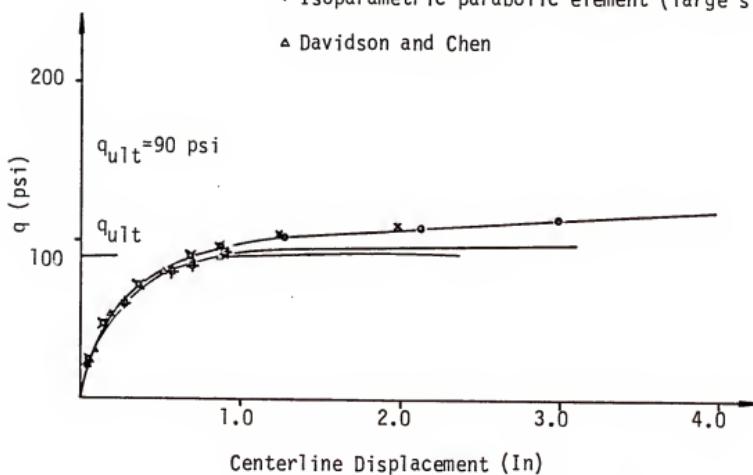


Figure 7-3 Footing response for Von Mises material

Drucker-Prager model:

$$E = 30,000 \text{ psi}$$

$$\nu = 0.3$$

$$c = 10 \text{ psi}$$

$$\phi = 20^\circ$$

Cap model:

$$E = 30,000 \text{ psi}$$

$$\nu = 0.3$$

$$c = 10 \text{ psi}$$

$$\phi = 20^\circ$$

$$R = 4.0$$

$$W = 0.003$$

$$D = 6.042 \times 10^{-5} \text{ psf}^{-1}$$

Results are presented in Figure 7-4 for the Drucker-Prager and Cap models. The limiting cases are compared with the collapse solution of Prandtl, Coulomb and Terzaghi. The numerical analysis is compared with that done by Mizuno [68] with the rectangular elements composed of the CST elements.

A comparison of the load-displacement relations predicted by the Drucker-Prager and Cap models shows that the Cap model predicts larger settlements at low stresses. This is a reasonable development since the Drucker-Prager model neglects the influence of the plastic volumetric compaction. Therefore, it is expected that the settlement prediction obtained using the Drucker-Prager model will be valid for only over-consolidated soils.

Author's Solutions:

- Isoparametric linear element (small strain)
- Isoparametric parabolic element (small strain)
- ×
- + Isoparametric linear element (large strain)
- + Isoparametric parabolic element (large strain)

Mizuno's Solutions:

- Isoparametric linear element (small strain)
- Isoparametric parabolic element (small strain)

(P) Prandtl -- $q_{ult} = 143$ psi
 (C) Coulomb -- $q_{ult} = 152$ psi
 (T) Terzaghi -- $q_{ult} = 175$ psi

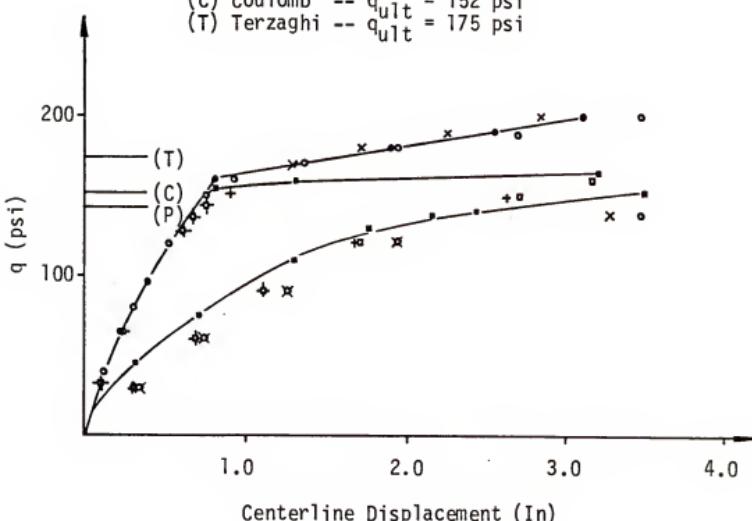


Figure 7-4 Footing response for Drucker-Prager and Cap material

The "locking" phenomenon is found again for the linear isoparametric formulation in both the Drucker-Prager model and the Cap model. The predictions with small and large deformation assumptions also yield similar results. The analyses for the Drucker-Prager model and the Cap model were performed with the same number of loading increments and a convergence tolerance of 10^{-8} .

Uniaxial Volumetric Compaction

The plastic soil models were further tested by considering a plane strain soil specimen under uniaxial strain. Both the Drucker-Prager model and the Cap model were again used in the analysis. The material constants are the same as those used by Baladi and Rohani [1]:

$$E = 841,396 \text{ psf}$$

$$\nu = 0.2763$$

$$c = 0$$

$$\phi = 49.093^\circ$$

$$R = 4.33$$

$$D = 6.781 \times 10^{-5} \text{ ft}^2/\text{lb}$$

$$W = 0.0075$$

The load-displacement curves are shown in Figure 7-5 for each model. The soil is loaded vertically up to 24 kips/ft² and then unloaded to zero under a static condition. The solid lines present the load-displacement curve predicted by the Drucker-Prager model. In this case, the load-displacement relationship is linearly elastic under the loading as well as unloading path. As can be seen from Figure 7-5b, the stress path in $(I_1, \sqrt{J_2})$ space for this case is a linear line within a failure envelope.

The curve connected with squares shows the behavior predicted by the Cap model under a loading condition. The load-displacement relationship is not linear from the beginning. This stems from the fact that a compaction of plastic volumetric strain caused by a hardening cap influences the behavior of the material under a low level I_1 . As the applied pressure becomes higher, however, the state of the compaction in the material approaches that of maximum compaction. Therefore, the slope of the load-displacement curve under a higher level of the applied load becomes gradually identical to that of the solid line predicted by the Drucker-Prager model. During loading (Figure 7-5c) the stress path in $(I_1, \sqrt{J_2})$ space moves within the failure envelope and is situated on the hardening cap. On the other hand, the stress path during unloading (Figure 7-5d) moves within the current elastic region while the hardening cap remains fixed in stress space. Consequently, the load-displacement curve shows linear elastic behavior until the stress path touches the failure envelope on the opposite side. Once the stress path reaches the failure envelope, a volume expansion or dilatancy develops. After the completion of the loading-unloading cycle, the residual compaction of 0.15 inches at the surface is predicted by the Cap model.

This problem was also analyzed by Mizuno [69], and identical results have been reported.

Test of Modified Stiffness Method in Unloading

As stated in Chapter 6, the modified stiffness method proposed by the author can automatically predict the unloading behavior. The validity of this method was tested by using the specimen shown in Figure 7-6a. The Drucker-Prager model with the following properties are used:

$$E = 500,000 \text{ psf}$$

$$\nu = 0.2$$

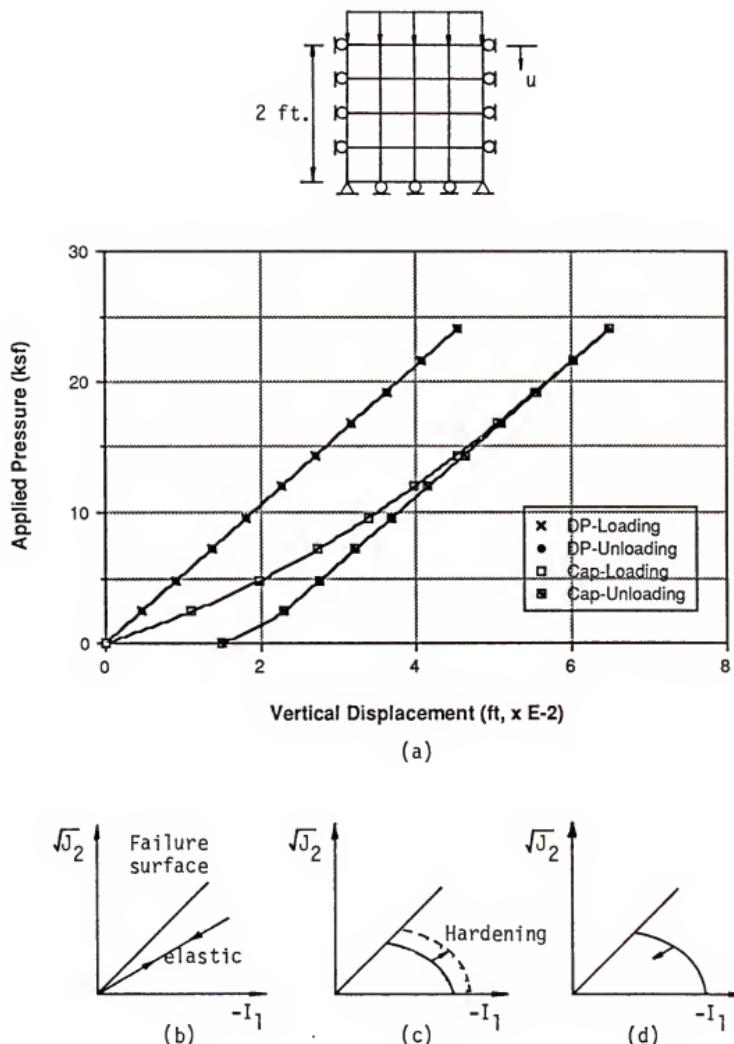


Figure 7-5 Uniaxial volumetric compaction

$$c = 500 \text{ psf}$$

$$\phi = 30^\circ$$

Two loading cases were analyzed. For the first case, a vertical compressive load of 1690 psf was applied to the top surface, while, for the second case, the load was applied as follows:

1. A vertical compressive load of 1340 psf was applied to the top surface.
2. The specimen was then unloaded by applying a vertical tensile load of 1340 psf.
3. Reloading was then performed by adding a compressive loading of 1500 psf.
4. The specimen was then again unloaded by applying a vertical tensile load of 1500psf.
5. Reloading was then again performed by adding a compressive loading of 1690 psf.

The two loading cases were first analyzed considering material nonlinearity only. The load-displacement curves are shown in Figure 7-6b for both cases. The unloading behavior is linearly elastic since the unloading path is parallel to the initial linear elastic loading path. Reloading first traces the unloading path and then follows the load-displacement curve predicated by the first case. The two loading cases were further analyzed considering both geometric and material nonlinearities. The results were shown in Figures 7-7., The obtained solutions indicate that the proposed "mixed" stiffness method can simulate unloading correctly.

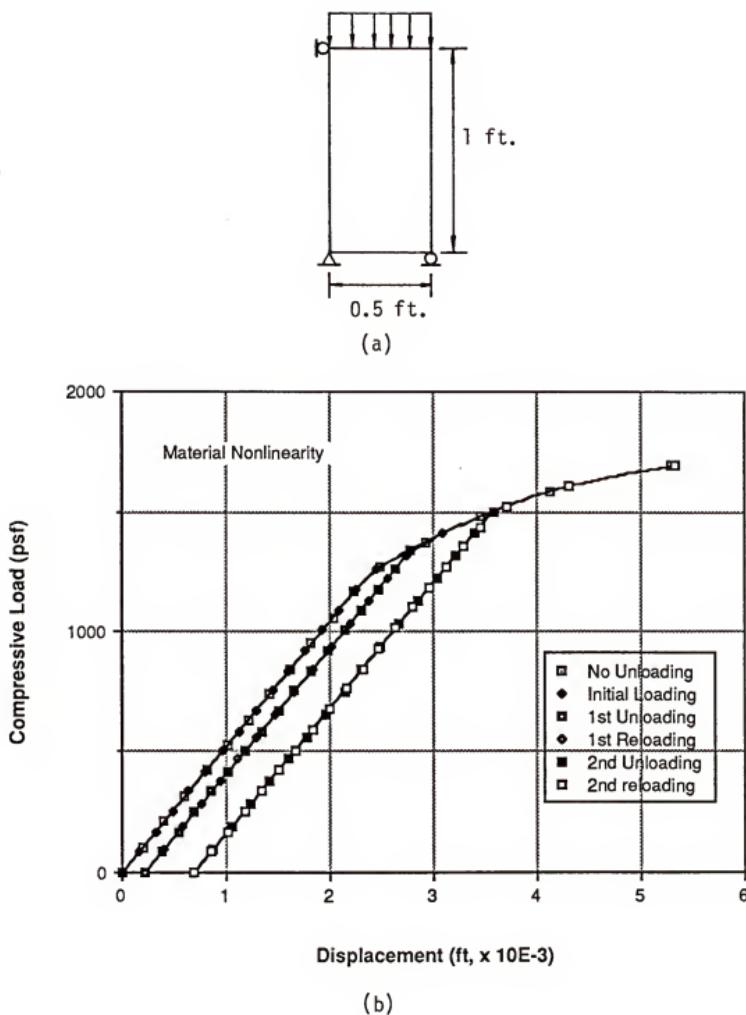


Figure 7-6 Loading-unloading load-displacement curves

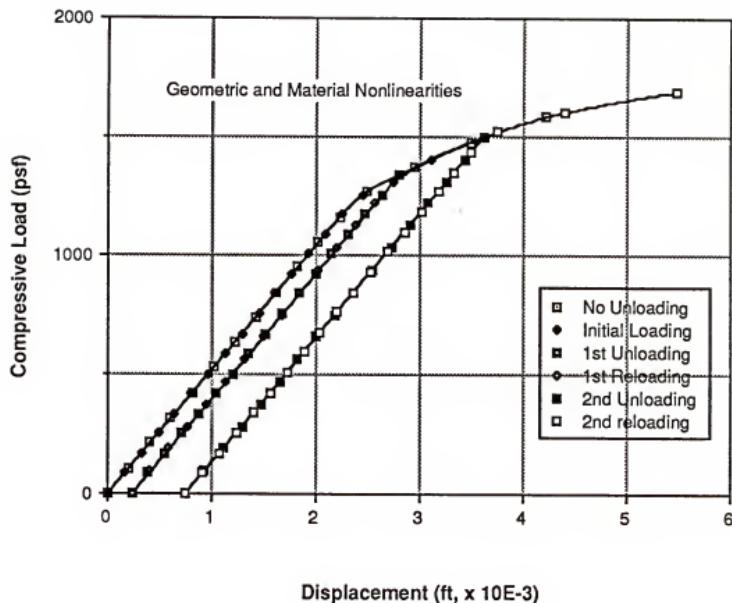


Figure 7-7 Loading-unloading load-displacement curves

One Dimensional Excavation

A one-dimensional plane strain problem for which the exact solution can be found by the theory of elasticity is analyzed [93]. The dimensions of the problem are given in Figure 7-8a and the material properties are

$$E = 10,000 \text{ t/m}^2$$

$$\nu = 0.2$$

$$\gamma = 1.0 \text{ t/m}^3$$

$$k_0 = 0.5$$

This problem was also solved by Sargand [81] by using a hybrid finite element method and by Mana [64] by using the displacement finite element method. Figure 7-8b and 7-8c compare different numerical results with the exact solution. It can be seen that the equivalent nodal forces and displacements computed by the author and Sargand using only one element yield exact solutions, whereas the displacement method used by Mana using eight elements failed to do so. Compared to the results by Mana, Sargand argued that the hybrid finite element method can provide better convergence for stresses and displacements than the displacement method. However, by using the excavation simulation techniques described in chapter 5, the displacement finite element method can provide the same accurate results.

Uniqueness Test for Excavation

In excavation, when a layer of soil medium with linear elastic properties is excavated in one step or several steps the behavior should be the same. This phenomenon was evaluated through the use of a plane strain problem analyzed by Clough [17]. Figure 7-9 shows the geometry of the problem. Element of unequal size and arbitrary shapes were included in the finite element mesh, and the excavation boundary consisted of

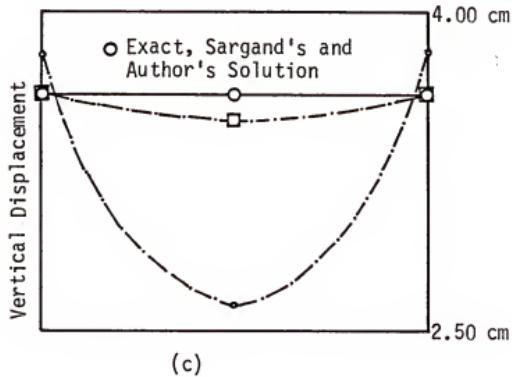
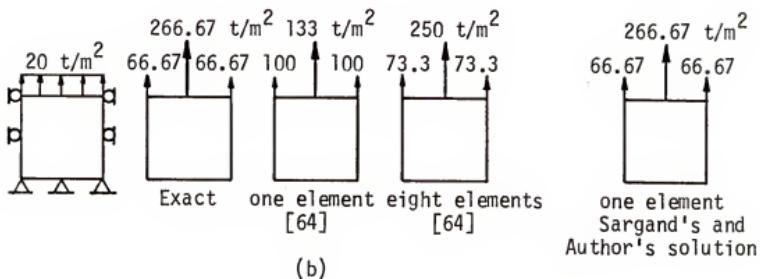
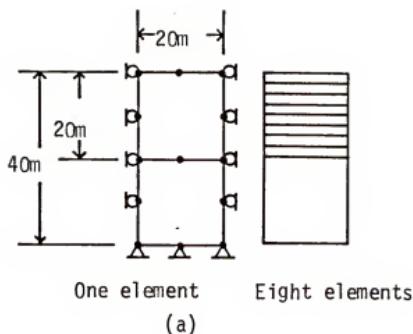


Figure 7-11 One-dimensional excavation; (a) Finite element representation; (b) Comparison of equivalent nodal forces; (c) Comparison of displacement

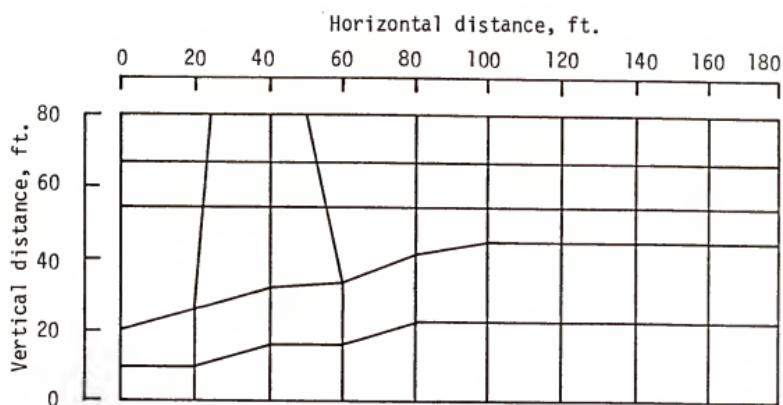


Figure 7-9 Finite element mesh for uniqueness test

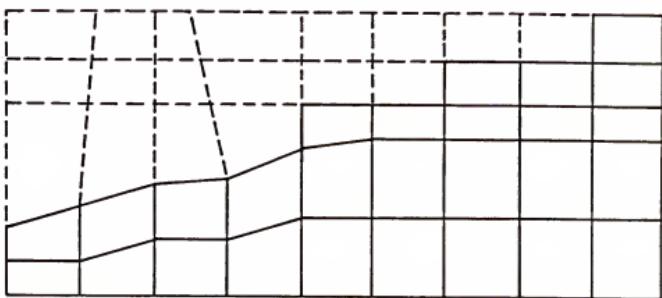


Figure 7-10 One-step excavation

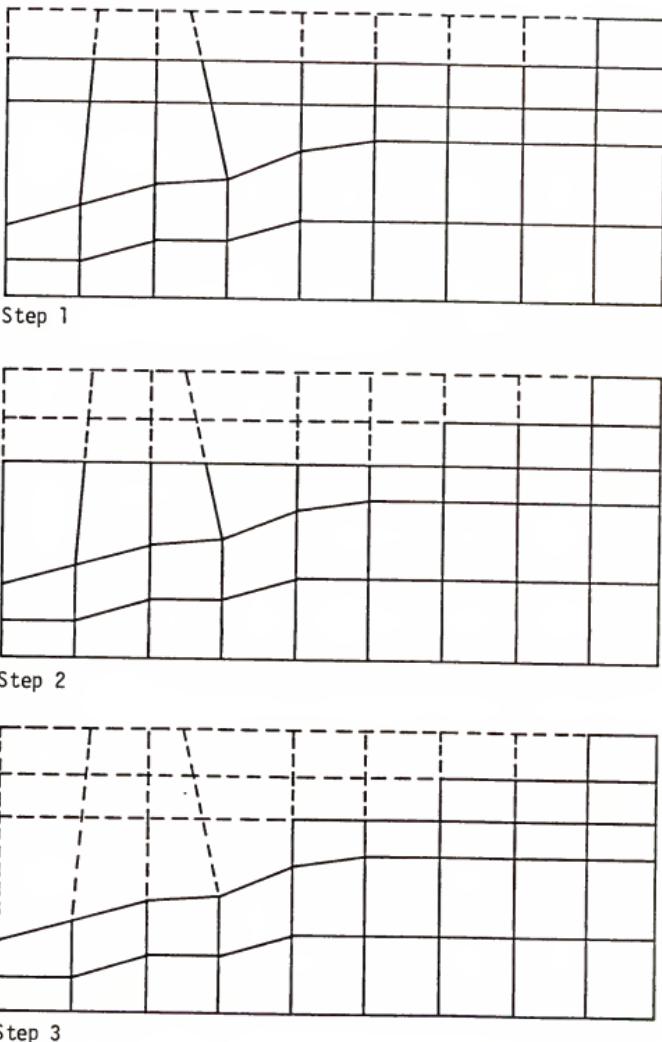


Figure 7-11 Three-step Excavation

TABLE 7-1 Displacements on excavated surfaces

FEMCON Solution

Nodes	1-Step Excavation				3-Step Excavation			
	X-Disp.	Y-Disp.	X-Disp.	Y-Disp.	X-Disp.	Y-Disp.	X-Disp.	Y-Disp.
4	.00000	1.1265	.00000	.26126	.00000	.48872	.00000	1.1265
10	-.0578	1.2712	-.0040	.32648	-.0154	.60824	-.0578	1.2712
16	-.1108	1.4733	-.0088	.43085	-.0309	.79976	-.1108	1.4733
22	-.1620	1.4296	-.0156	.44516	-.0546	.82704	-.1620	1.4296
27	-.1648	1.5363	-.0209	.71133	-.0629	1.3154	-.1648	1.5363
28	-.2314	1.1928	-.0254	.49889	-.0871	.92214	-.2314	1.1928
33	-.0910	1.3167	-.0276	.70473	-.0581	1.2586	-.0910	1.3167
38	.01344	1.0776	-.0016	.82926	.04094	1.0906	.01344	1.0776
39	-.1228	.94503	-.0341	.67725	-.0817	.9579	-.1228	.94503
44	.04766	.79068	.02410	.74574	.07207	.81431	.04766	.79068
49	-.0096	.31842	-.0130	.36061	-.0098	.34211	-.0096	.31842
50	-.0879	.36368	-.0083	.40428	.00121	.38710	-.0879	.36368

unit: ft.

SOIL-STRUC Solution

Nodes	1-Step Excavation				3-Step Excavation			
	X-Disp.	Y-Disp.	X-Disp.	Y-Disp.	X-Disp.	Y-Disp.	X-Disp.	Y-Disp.
4	.00000	1.1139	.00000	.25952	.00000	.48137	.00000	1.1144
10	-.0834	1.2794	-.0034	.32497	-.0126	.60362	-.0817	1.2797
16	-.1147	1.4849	-.0091	.43068	-.0331	.80246	-.1103	1.4866
22	-.1762	1.4390	-.0166	.44612	-.0592	.83253	-.1686	1.4412
27	-.1446	1.5362	-.0242	.71293	-.0736	1.3220	-.1331	1.5458
28	-.2289	1.1907	-.0269	.50055	-.0883	.93058	-.2227	1.1936
33	-.0697	1.3229	-.0300	.70863	-.0498	1.2756	-.0607	1.3226
38	.03251	1.0738	-.0041	.83417	.06674	1.1015	.03960	1.0827
39	-.1182	.94464	-.0316	.68467	-.0758	.96620	-.1145	.94656
44	.06490	.78946	.03457	.75283	.09363	.81412	.06780	.78939
49	.01354	.30185	.01029	.34886	.01098	.32305	.01340	.30260
50	-.0167	.35286	-.0148	.39852	-.0050	.37494	-.0150	.35370

unit: ft.

inclined as well as horizontal and vertical segments. The material is assumed to be linear elastic with the following properties:

$$E = 100,000 \text{ psf}$$

$$r = 100 \text{pcf}$$

$$\nu = 0.3$$

$$K_0 = 0.43$$

The problem involves excavating a block of material by using one step and three steps as shown in Figure 7-10 and 7-11.

The numerical results in terms of displacements from one-step and three-step excavations are identical as is shown in Table 7-1. This indicates that the excavation simulation technique employed can provide unique solutions independent of the number of excavation steps. Table 7-1 also compares with the displacements obtained from Program SOIL-STRUC by Clough. The comparison appears to be fairly good. However, an exact correspondence in displacements can not be obtained using SOIL-STRUC.

Construction Sequences

The capabilities of the computer code, FEMCON, to model construction sequences were examined using a simple sixteen element mesh. Figure 7-12a shows the geometry and dimensions for the test problem. The geometry was chosen such that the results are the same for all vertical sections. A plane strain idealization was assumed. A linear elastic constitutive model was utilized in order that the results might be compared to closed-form solutions. The material properties used in the problem are

$$E = 1.0 \times 10^6 \text{ psf}$$

$$\nu = 0.2$$

$$k_0 = 0.25$$

$$\gamma = 122.4 \text{ pcf}$$

$$\gamma_w = 62.4 \text{ pcf}$$

The simulation of construction sequences for this problem is illustrated in Figure 7-12.

Insitu Stress

The mesh shown in Figure 7-12a was analyzed to find the proper insitu stresses for use in the dewatering simulation. Plots of the computed variation of the vertical and horizontal stresses with depth are given in Figure 7.13a. The computed values are exact in that the vertical stresses are equal to the Buoyant unit weight of the material multiplied by the depth and the horizontal stresses are equal to the coefficient of lateral earth pressure multiplied by the vertical stresses.

Dewatering

Using the insitu stress results from above, the dewatering sequence was simulated in which the water table is lowered from level A to level B as shown in Figure 7-12a and 7-12b. The computed stresses as functions of depth are shown in Figure 7.13b. The net effect of lowering the water table was in accordance with what was expected. Lowering the water table removes the Buoyant forces in the dewatered region and increases its effective weight by an amount equal to the unit weight of water, thereby increasing the effective stress in the region.

Excavation

The simulation of excavation was performed by removing the top two layers of elements from the mesh as shown in Figure 7-12c. As can be seen in Figure 7-13c, the

stresses are decreased by an amount equal to the height of the excavated layers multiplied by its density.

Embankment

The simulation of embankment was purposely performed by putting the excavated material back to the top two layers of the mesh. As can be seen in Figure 7-13d, the stresses are increased by an amount equal to the height of the embankment layers multiplied by its density. However, due to the linear material properties, the stresses after embankment are identical to those after dewatering.

Reestablishment of Water Table

After the embankment was completed, the water table was raised to its original height. This is shown in Figure 7-12e. Contrary to dewatering, as can be seen in Figure 7-13d, raising the water table will decrease the effective stresses. Again, due to the linear material properties, the results in Figure 7-13e are the same as those in Figure 7-13a of the insitu stress case.

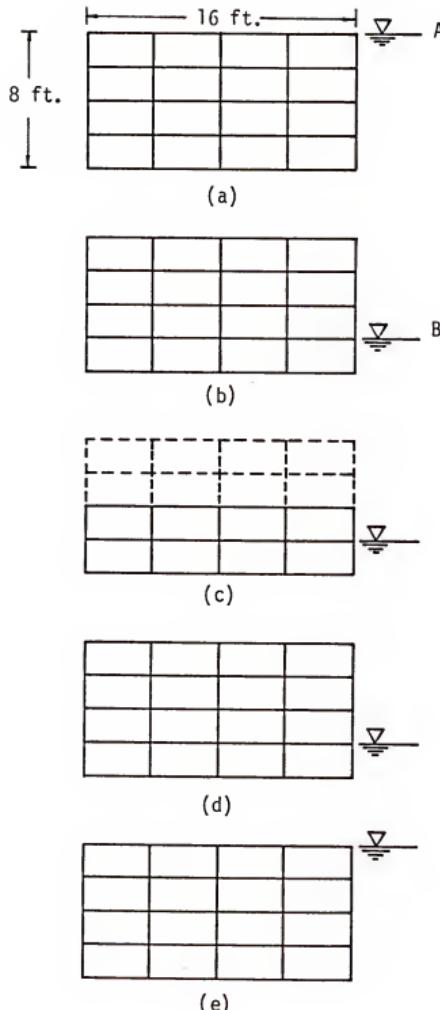


Figure 7-13 Construction sequences; (a) Insitu; (b) Dewatering; (c) Excavation; (d) Embankment; (e) Reestablishment of water table

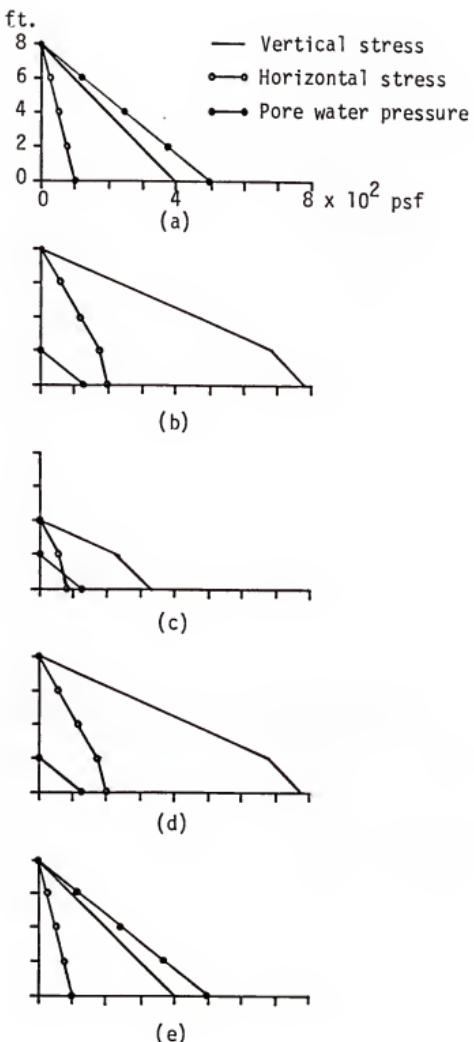


Figure 7-13 Stress results; (a) Insitu; (b) Dewatering; (c) Excavation;
(d) Embankment; (e) Reestablishment of water table

CHAPTER 8

DISCRETE ELEMENT ANALYSIS

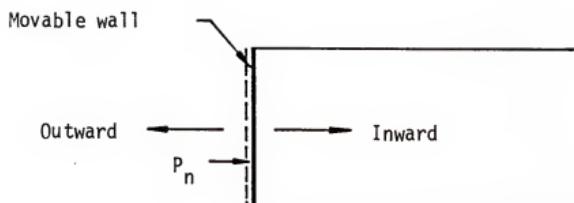
In this chapter, the discrete element method will be used to analyze the U-frame structures. Haliburton [45] and Hays [46] had demonstrated the use of the discrete element method for analysis of soil-structure interaction problems including piles, braced excavation and sheet piles. The nonlinear discrete element program FRAME54, developed by Hays, is used in this study. In this analysis, construction sequence was not considered.

The discrete element method uses nonlinear Winkler-type springs ($q-w$ curves or soil response curves) to represent the soil medium. The $q-w$ curves for the earth retaining structures can be determined experimentally as done by Sheriff and Fang [85], or obtained empirically as shown by Haliburton [45]. Haliburton defined the $q-w$ curves by using the Rankine theory and the empirically determined values of soil modulus. In this chapter, these curves are generated numerically using the finite element program, FEMCON, and a discussion on them is also presented.

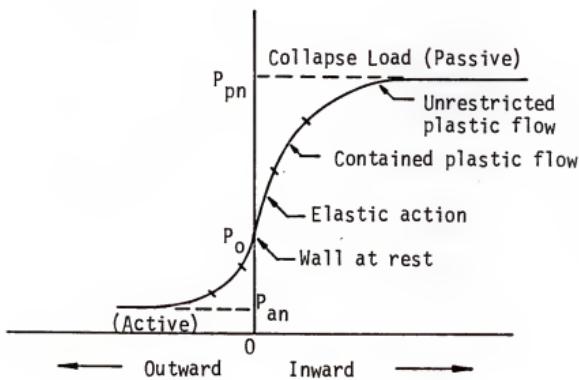
After defining the $p-w$ curves, the U-frame structures are analyzed using the discrete element method as well as the finite element method. The discrepancies obtained from the numerical results by the two methods are explained and a method is suggested to bring the two solutions closer.

Soil Response Curves by Finite Element Method

Before trying to generate $q-w$ curves by the finite element analysis, it might be useful to review the relation between wall movement and earth pressure. This relation is shown qualitatively in Figure 8-1. The wall is constructed so that it can be held in a



(a)



(b)

Figure 8-1 Results of retaining wall tests; (a) Vertical section through bin;
 (b) Load-displacement relationship

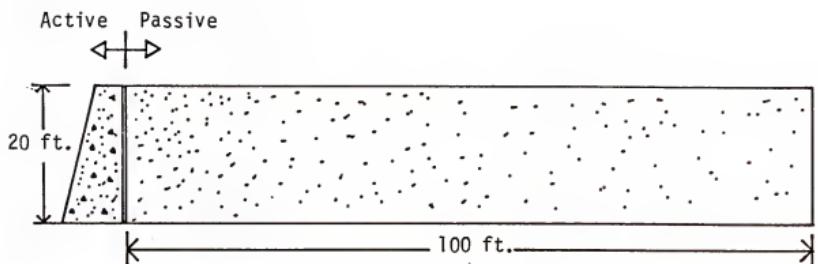
fixed position or moved inward or outward. The wall is initially at rest and held by a force $P=P_0$. As the force P is reduced, the wall will be forced outward due to the weight of the soil. As P is gradually reduced, the soil initially undergoes elastic deformation, then elastic-plastic deformation and finally, uncontained plastic flow and thus defines the active collapse load, P_{an} . On the other hand, if the force P is increased from P_0 , displacement occurs and failure results as the passive collapse load P_{pn} is approached.

Having reviewed the load-displacement relationship, it is now possible to construct $q-w$ curves using the finite element method by moving the wall inward and outward. As mentioned previously, the discrete element method of analysis makes use of the well-known Winkler assumption concerning independent behavior of adjacent soil layers. Because of this assumption the effect of possible shearing in soil and between wall and soil is not considered. It is interesting to note here that this is the same assumption made in the Rankine earth pressure theory. To achieve this assumption, the rigid wall and backfill system shown in Figure 8-2a is used. Beginning from the initial at-rest pressure conditions, the wall was moved towards or away from the backfill in a series of increments, until the passive or active earth pressure condition was reached. Figure 8-2b shows the mesh used. In order to simulate the incremental displacements, bar elements with a very high stiffness were used. The effect of wall displacement can be accomplished by applying a load equal to the value of the bar stiffness times the displacement to the nodal points that connect bars and soil elements. It should be mentioned here that the results obtained from the finite element analysis are only as valid as the stress-strain parameters used in the analysis. The Drucker-Prager soil model is used in this chapter.

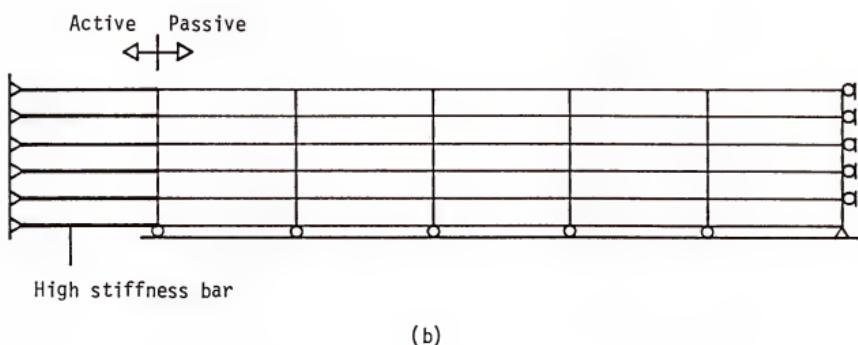
Two cases are analyzed. The first case deals with a cohesionless soil with the following properties:

$$E = 4.32 \times 10^6 \text{ psf}$$

$$\nu = 0.30$$



(a)



(b)

Figure 8-2 Retaining wall-backfill system to generate q-w curves;
(a) retaining wall and backfill; (b) Finite element mesh

$$\gamma = 120 \text{ pcf}$$

$$K_0 = 0.43$$

The second case deals with a cohesive soil with the following properties:

$$E = 4.32 \times 10^6 \text{ psf}$$

$$v = 0.30$$

$$\gamma = 150 \text{ pcf}$$

$$K_0 = 0.43$$

$$c = 360 \text{ psf}$$

$$\sigma = 20^\circ$$

The obtained load-displacement curves and p-w curves are shown in Figures 8-3, 8-4 , 8-5, 8-6 for the two cases. Due to the same assumption used in the finite element analysis and the Rankine earth pressure theory, the obtained results can be verified by the Rankine earth pressure theory. Thus, here the Rankine earth pressure theory is numerically verified using the plasticity Drucker-Prager model.

Analysis of U-frame Structures

The p-w curves shown in Figure 8-6 were used in this section to analyze a retaining wall and two U-frame structures. These structures are solved by the discrete element method and the finite element method. The concrete is used as the structural material and has the following properties:

$$E = 4.32 \times 10^8 \text{ psf}$$

$$v = 0.2$$

$$\gamma = 150 \text{ pcf}$$

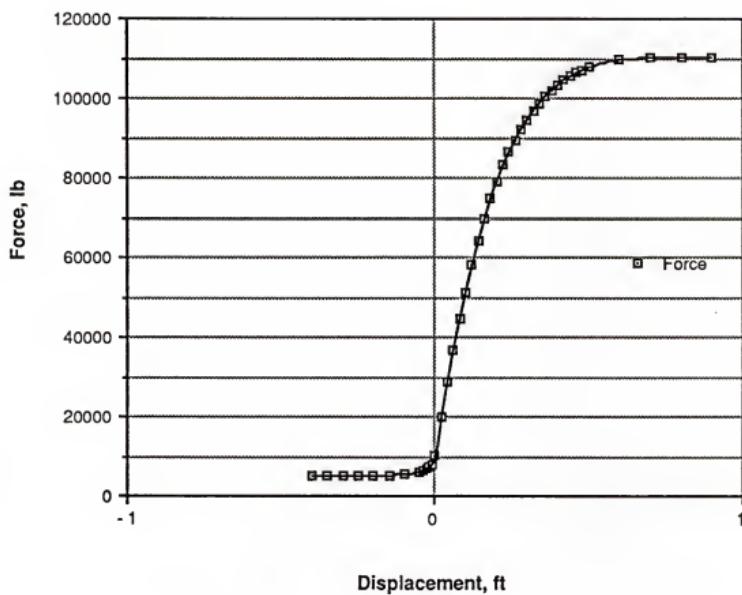


Figure 8-3 Load-displacement (P-W) curve for a cohesionless soil

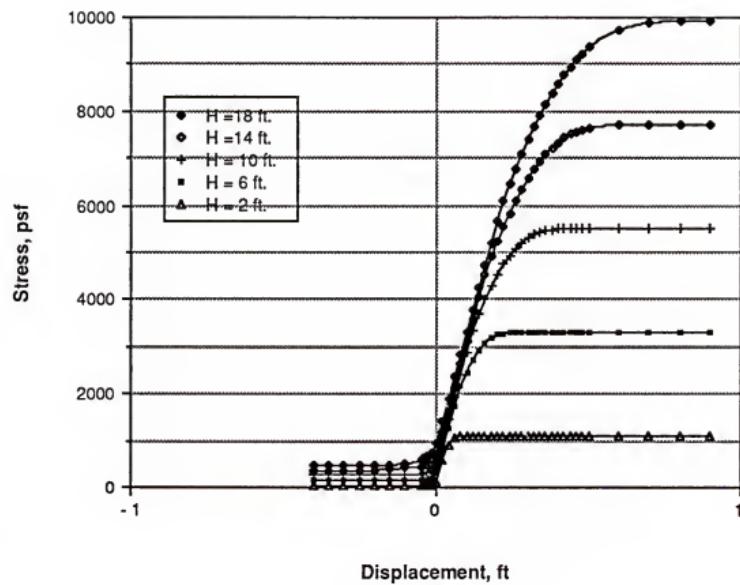


Figure 8-4 Soil-response ($q-w$) curve for a cohesionless soil

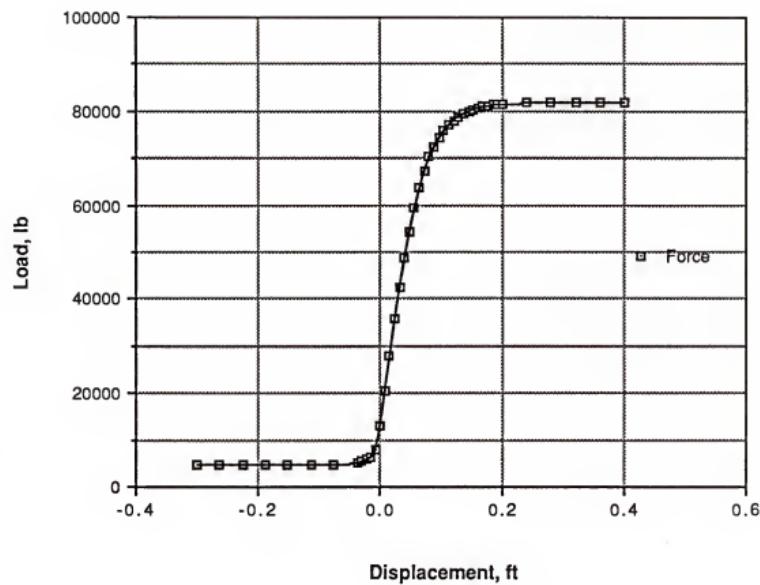


Figure 8-5 Load-displacement (P-W) curve for a cohesive soil

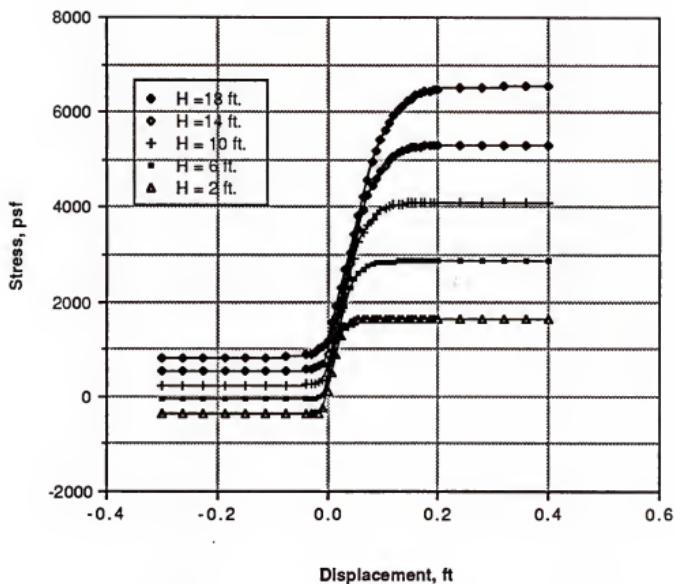


Figure 8-6 Soil response ($q-w$) curve for a cohesive soil

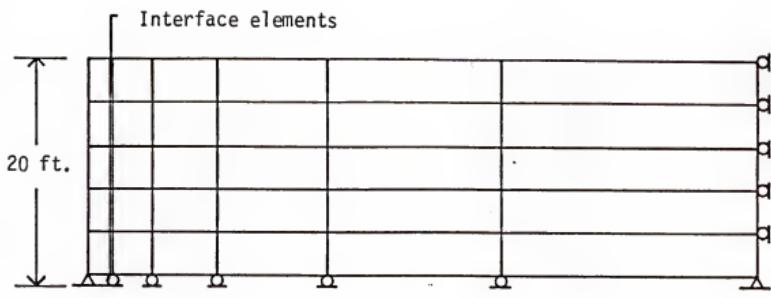
The retaining structure and its discrete element and finite element representations are shown in Figure 8-7. In the finite element mesh, the interface elements behind the wall are added to consider the relative displacement between the wall and the backfill, and a very small shear modulus is assigned to simulate the frictionless smooth wall behavior. The obtained results are shown in figure 8-8a. Clearly, it can be seen that the discrete element analysis predicts a much larger wall deflection. This is due to the fact that the p-w curves used neglect the shear stresses between soil layers. However, when the soil exerts its self weight on the flexible wall, the shear stresses developed due to the self weight of the soil tend to prevent the soil from moving laterally. In order to consider the shear effect, the p-w curves are modified by using a larger soil modulus. It is found that in this case the discrete element solution can be adjusted approximately to the finite element solution if a soil modulus about three times its original value is used. The adjusted solution is shown in Figure 8-8b.

The next structure analyzed is an unsymmetrical U-frame structure on a rigid base. Its finite element and discrete element representations are shown in Figure 8-9. The same interface element is used in the finite element analysis and the interaction between the structure and the rigid base is assumed to be frictionless. The obtained results using the original soil modulus in Figure 8-6 are shown in Figure 8-10a. It is found that in this case the discrete element solution can be adjusted approximately to the finite element solution if a soil modulus about two times its original values is used. The adjusted solution is shown in Figure 8-10b.

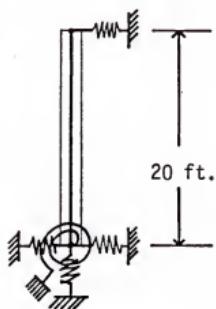
The final structure analyzed is an unsymmetrical U-frame structure in soil medium. Its finite element and discrete element representations are shown in Figure 8-11. The same interface element is used. For the discrete element analysis, in this case, it is necessary to generate the soil response curve (Winkler spring) under the U-frame base slab. This is done by using the structure shown in Figure 7-5. The obtained

results using both lateral and base soil response curves are shown in Figure 8-12. For this case, however, the author found that it is very difficult to adjust the discrete element solution to the finite element solution. This is due to the added complexities caused by the base Winkler springs. It is noted here that in generating the Winkler spring curves the shearing between the springs is ignored and the confining pressures acting between them are very difficult to simulate.

From the study of the above three structures, to obtain accurate discrete solutions, it would be necessary to generate soil response curves that include the shearing effects. These curves, however, at this time can not be generated by FEMCON. This is due to the lack of the plasticity interface element models to simulate the nonlinear shearing behavior between two different materials. These types of models are rarely found in the literature and this is a good area for future research.



(a)



(b)

Figure 8-7 Retaining wall deflection; (a) Finite element model;
(b) Discrete element model

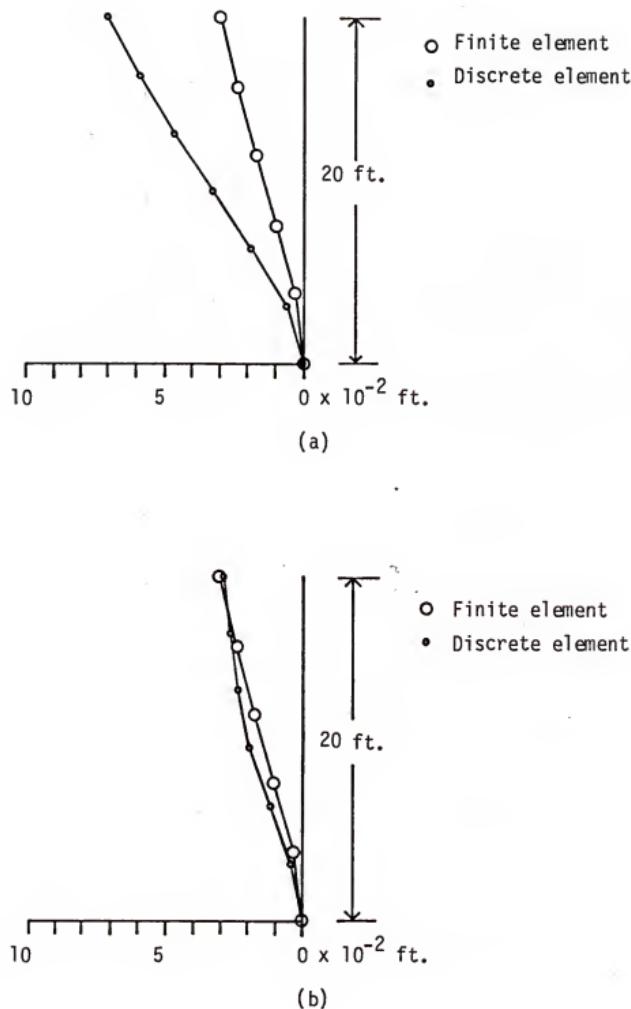


Figure 8-8 Retaining wall deflection; (a) Original solutions; (b) Modified solutions

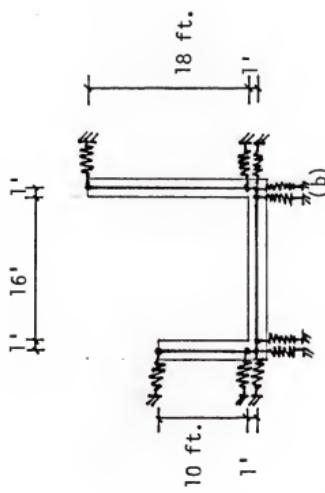
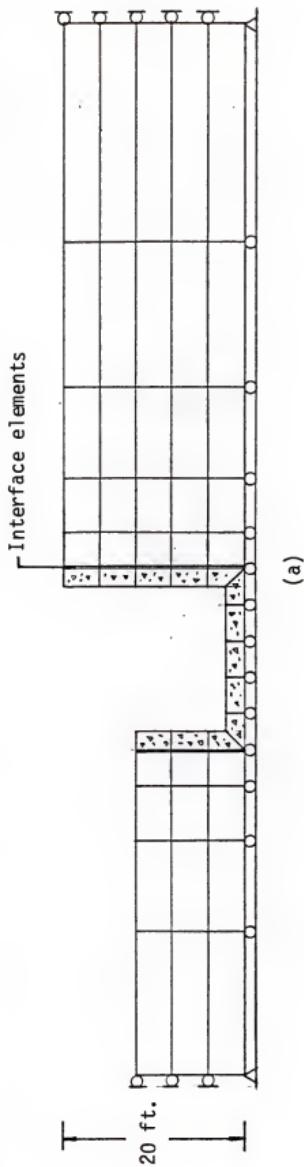


Figure 8-9 U-frame on a rigid base; (a) Finite element model; (b) Discrete element model

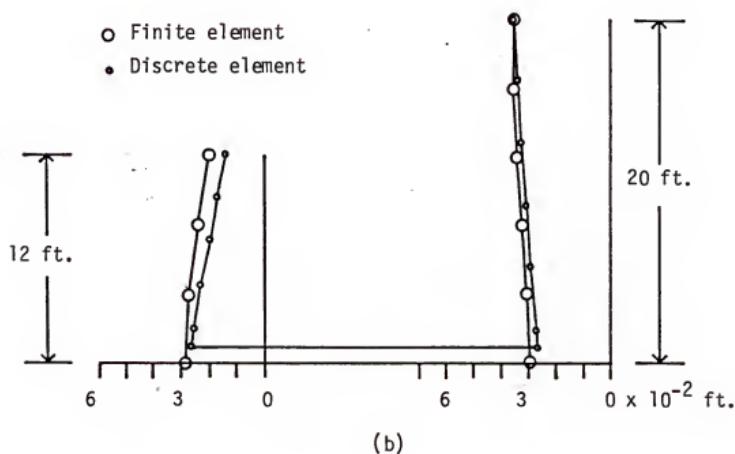
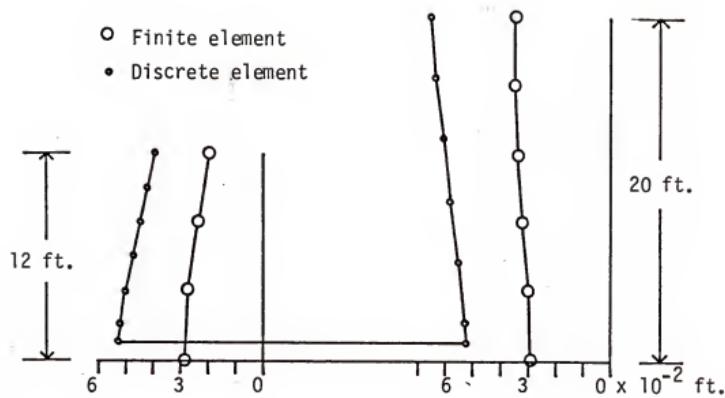


Figure 8-10 U-frame wall deflection; (a) Original solutions; (b) Modified solutions

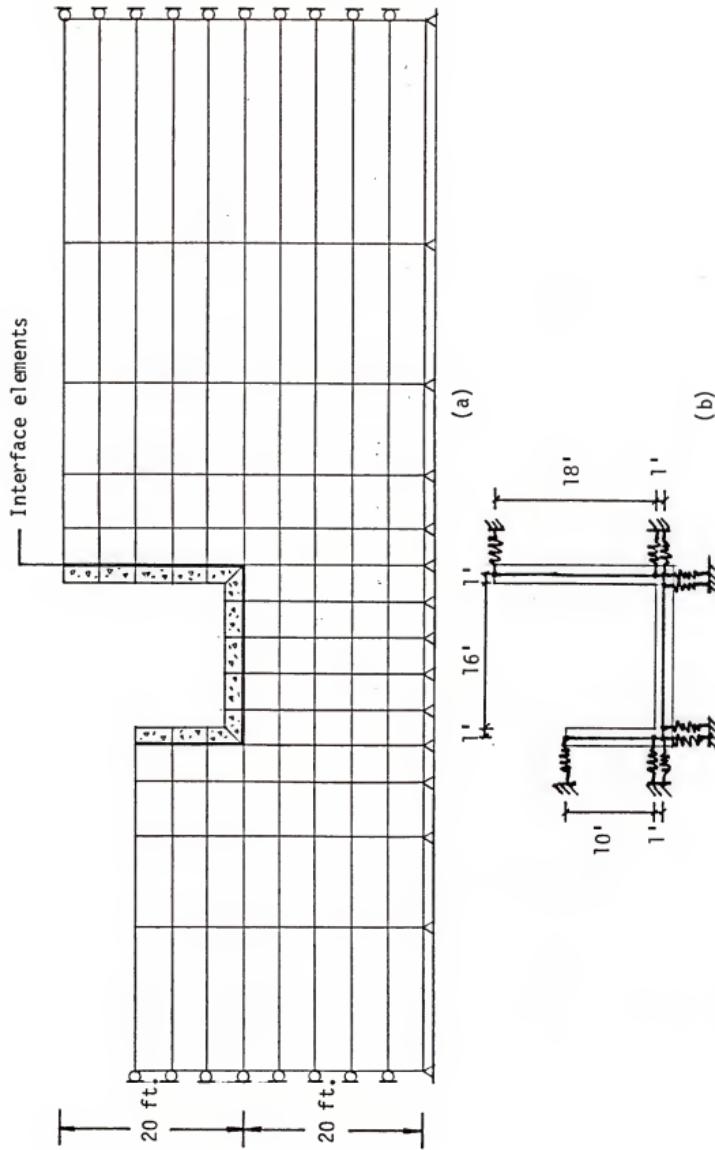


Figure 8-11 U-frame in soil medium; (a) Finite element model; (b) Discrete element model

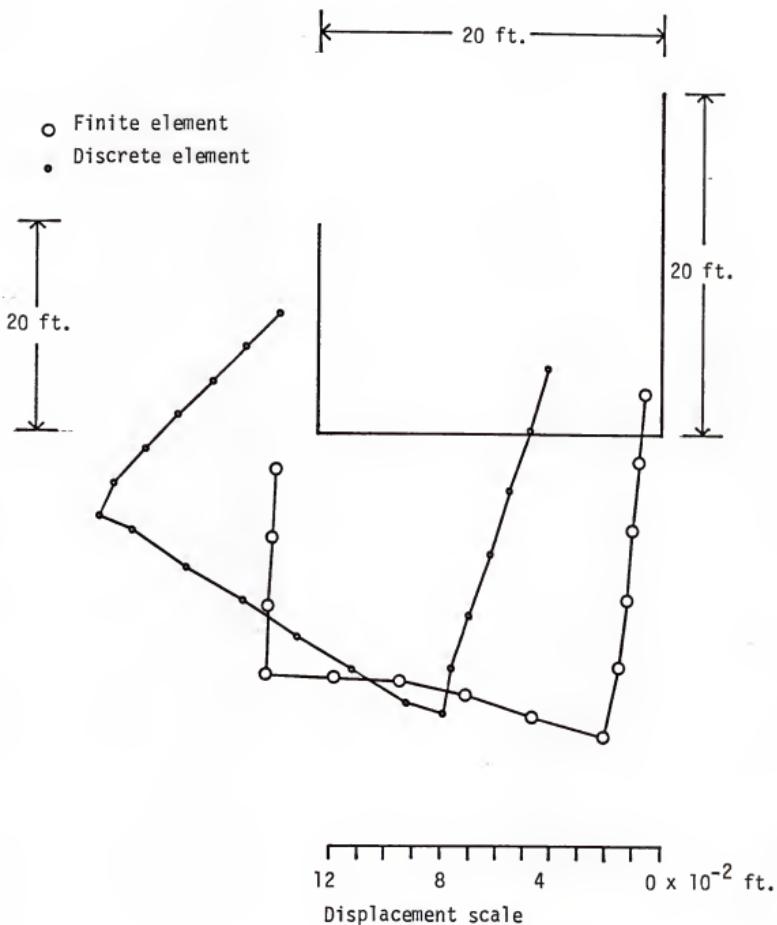


Figure 8-12 U-frame displacements

CHAPTER 9

FIELD PROBLEMS

In this chapter, two field problems were analyzed in order to evaluate the proposed formulation with regard to more realistic problems. The problems are: (1) Port Allen lock, and (2) braced excavation in soft clay. The predictions from the present study are compared with field measurements and with previous FEM analyses. The material properties and the details of the field problems are adopted from other investigators.

Port Allen Lock

Introduction

The Port Allen Lock, located on the right bank of the Mississippi River opposite Baton Rouge, La., is a reinforced concrete U-frame lock structure with a navigation chamber 84 ft wide, 1,200 ft long and 68 ft high. A cross section of Port Allen Lock is shown in Figure 9-1.

Recognizing the need to clarify the nature of the earth pressure distribution for future U-frame lock design, the U. S. Corps of Engineers undertook a very comprehensive instrumentation program for Port Allen Lock. This instrumentation program was carried out by the U. S. Army Engineers Waterways Experiment Station. Details of the instrumentation have been described by Sherman and Trahan [86]. A nonlinear finite element analysis of this problem was reported by Clough and Duncan [17] using the variable moduli (Duncan-Chang) soil model. The results from the present analysis, which considers both material and geometric nonlinearities, are

compared with those from the previous finite element analysis and the instrumentation program. For a consistent comparison the elastic-plastic material parameters were obtained based on experiments and studies done by Sherman and Trahan, and Clough and Duncan. The initial soil profile is shown in Figure 9-2.

Problem Description

The incremental finite element analyses of Port Allen Lock were performed by simulating each of the actual construction operations in one or more analytical sequences involving changes in loading and mesh geometry. Each of the various loading sequences was simulated by one of the techniques described in chapter 5. Beginning from the initial condition of the site with insitu stresses and water pressures, these steps progressed through excavation and dewatering, placement of concrete and backfill, reestablishment of normal ground water conditions, and filling of the lock with water. A schematic representation of finite element simulation of Port Allen Lock is shown in Figure 9-3. The finite element mesh, which is shown in Figure 9-4, was used throughout these analyses of various sequences. Depending on the sequences represented, the elements in the mesh were assigned properties representative of the natural soils; air (after excavation); or concrete or backfill soil (after construction).

The insitu soil stresses and water pressure employed at the beginning stage of these analyses were calculated using the unit weights of soils, at-rest pressure coefficients, and the ground water table shown in the initial soil profile in Figure 9-2. Excavation was simulated in three operations as shown from Figure 9-3b to 9-3d. During the second and third operations, dewatering was included. The placement of concrete in the base slab and walls, the placement of backfill, and the reestablishment of normal groundwater conditions were analyzed in a series of increments as shown from Figure 9-3e to 9-2k. Subsequently, as shown by the final operation in Figure 9-3l, filling of the lock with water was also simulated. The sequence of operations represented

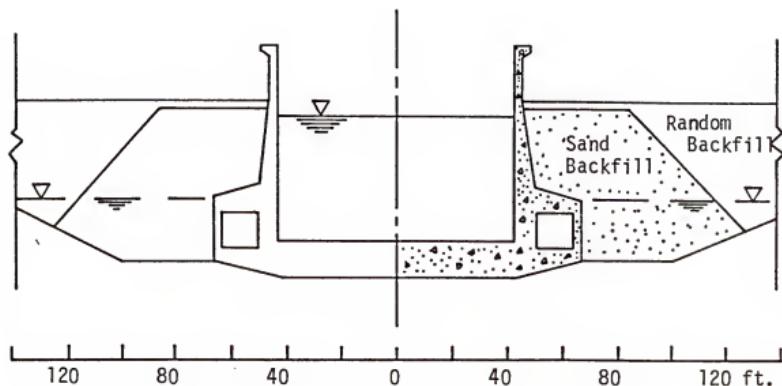


Figure 9-1 Cross section of Port Allen Lock

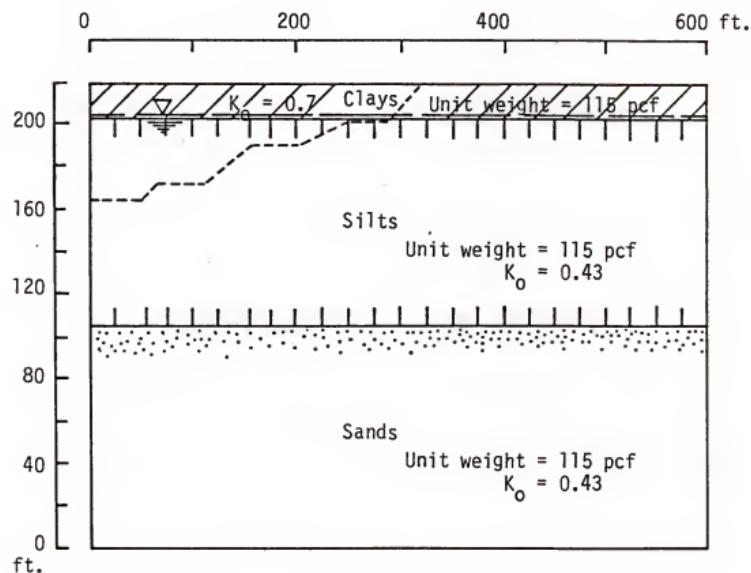


Figure 9-2 Initial soil profile for Port Allen Lock site

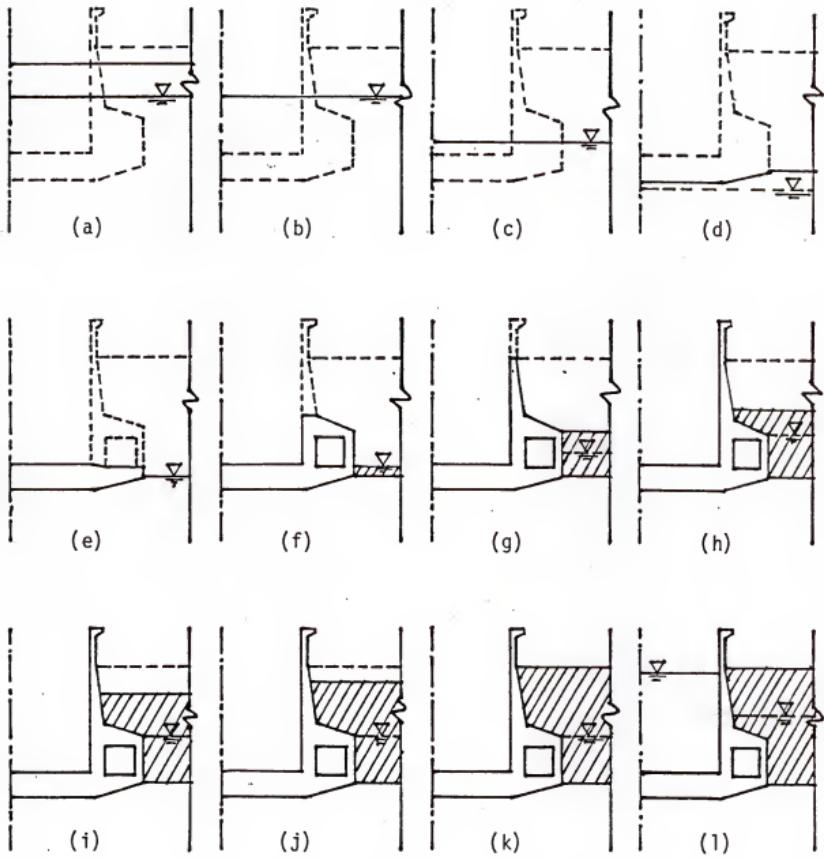


Figure 9-3 Schematic representation of finite element simulation of Port Allen Lock

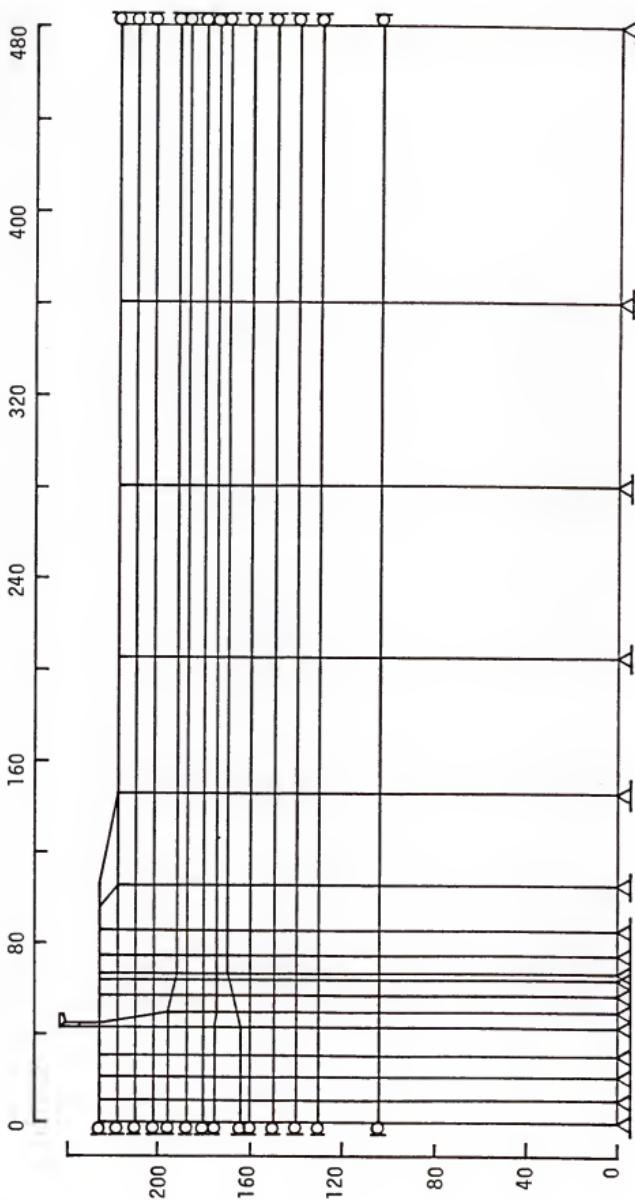


Figure 9-4 Finite element mesh employed for incremental analyses of Port Allen Lock

TABLE 9-1

Material Properties for Incremental Analyses of Port Allen Lock

Location	Soil type	E (psi)	v	γ (pcf)	K_o	c (psf)	ϕ degrees
Foundation	Silt	8,700	0.30	115.00	0.43	40.0	33.0
	Sand	10,600	0.30	115.00	0.43	0.00	40.0
Backfill	Sand	3,000	0.30	115.00	0.43	0.00	40.0
	Clay	1,500	0.30	115.00	0.43	80.0	26.0

Interface element properties: $E = 3000.0$ psi

v = 0.30

 $G = 1000.0$ psiConcrete properties: $E = 3.0 \times 10^6$ psi

v = 0.20

 $\gamma = 150.0$ pcf

in these analyses was chosen to correspond as closely as possible to the sequences of operations involved in construction of the lock.

The soil behavior was simulated as elastic-plastic using the Drucker-Prager model. Thin-layer elements were employed to represent the interface between the soil and concrete while two-dimensional quadrilateral elements used by Clough and Duncan were employed to represent the soil and concrete. The material properties used for the analysis are shown in Table 9-1. The concrete and interface element materials were assumed to be linear elastic.

Discussion of Results

Excavation Rebound. The variation of the calculated rebound of the center line of the excavation at the bottom of the base slab over a period of time is shown in Figure 9-5. Approximately 50% of the rebound occurred during the first increment because in the second and third increments dewatering served partially to counteract the effects of excavation. The only value of observed rebound reported by Sherman and Trahan was that for the end condition, shown in Figure 9-5. This value at the center line was 0.29 ft and at the edge of the excavation was 0.26 ft. The corresponding values calculated by the author were 0.30 ft and 0.26 ft and by Clough and Duncan were 0.22 ft and 0.21 ft. The differences between two finite element analyses are probably due to the different excavation simulation techniques and soil models used. The technique used by the author gives results closer to the field observation.

Settlement. The variation of the settlements of the center line of the lock with time is shown in Figure 9-5. From May 1958 to March 1959 the three sets of settlements agreed very closely. From this time to about December 1959, the water pressures on the base of the lock increased because the backfill was maintained in a saturated condition. Also, during this period, little weight was being added to the lock as the backfill was not yet being placed over the culvert. For these reasons, the observed

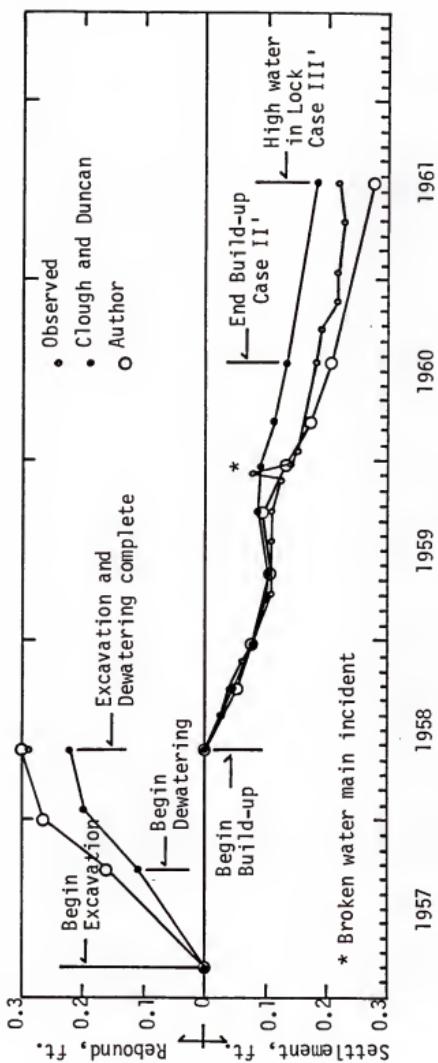


Figure 9-5 Variation of observed and calculated rebounds and settlement with time

settlements were small and the calculated settlements decreased slightly until September 1959. After that, the backfill was placed over the culvert and the calculated settlements increased gradually.

The final settlement of the lock at the end of construction before the lock filled with water (Case II') is shown in Figure 9-5. The calculated settlement was 0.21 ft and the observed value was 0.19 ft. Filling of the lock with water produced additional calculated settlement of 0.06 ft as opposed to an observed additional settlement of 0.03 ft as shown in Figure 9-5. The total calculated settlement for this condition (Case III') was 0.27 ft as compared to the observed value of 0.22 ft. It can be seen that the present analysis, in general, provides more accurate results than that by Clough and Duncan.

The reason for predicting more additional settlements for Case II' and III' may be due to that the elastic soil modulus used is a little lesser and the soils under the base slab have already yielded fully.

Structural Deflections. Figures 9-6 and 9-7 show structural deflections for case II' and III' respectively. It can be seen that the calculated wall deflections agree closely with field observations. It is interesting to see that after filling the lock with water and raising the water level outside of the lock the wall deflects inward a little.

The calculated base slab deflections, however, were inconsistent with those observed. At the outer edge the base slab deflects more for case II' and less for case III'. The reason for the deviation of the calculated and observed results at this location for case II' may be that the elastic soil modulus used under the base slab is slightly small and the backfill above the culvert causes the soil beneath it to yield fast and then settle more. For case III', the deviation may be due to the fact that the loading by filling of the lock with water and raising water level outside of the lock causes the base slab to deflect more at the center than at the outer edge. This is also the reason why the outward wall deflections for case III' are less than for case II'.

The calculated structural deflections, except the base slab deflection for case II', are also generally in agreement with those by Clough and Duncan. The difference in the case II' base slab deflection may be due to the different soil models used.

Effective Earth Pressures. The values of effective earth pressures given by the author were obtained from the interface elements. Figures 9-8 and 9-9 show the effective earth pressures acting on the lock for cases II' and III' respectively. Note that there is, in general, a good agreement between the calculated and observed earth pressures.

Variation of the effective earth pressure with time for a point at the center line of the base slab and a point on the upper lock wall slightly above the culvert are shown in Figure 9-10. The agreement between the three sets of values may be seen to be quite good. It can be seen that the calculated and observed effective earth pressures on the base showed an initial steady increase, but after the early part of 1959, there was a decrease in both the observed and measured values of effective earth pressure. This decrease was due to the increase in uplift pressures on the base slab, which was caused by raising the water table. Subsequently, the effective earth pressures increased again when the backfill was placed over the culvert.

The results of these nonlinear incremental analyses are in good agreement with the observed behavior. The observed rebounds during excavation, settlement during construction, earth pressures, and wall deflections agree closely with the calculated values. It may be concluded, therefore, that these analyses provide an effective means for analyzing complex soil-structure interaction problems and would provide a suitable analysis procedure for use in design studies for future U-frame locks.

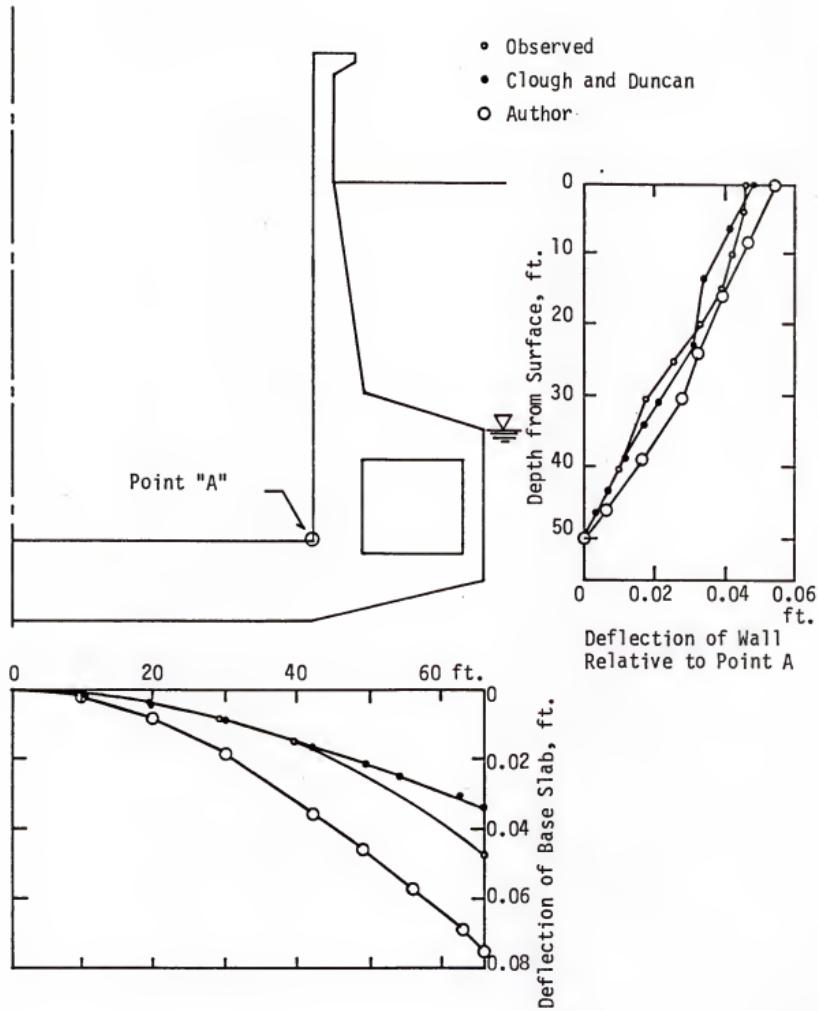


Figure 9-6 Structural deflections for case II'

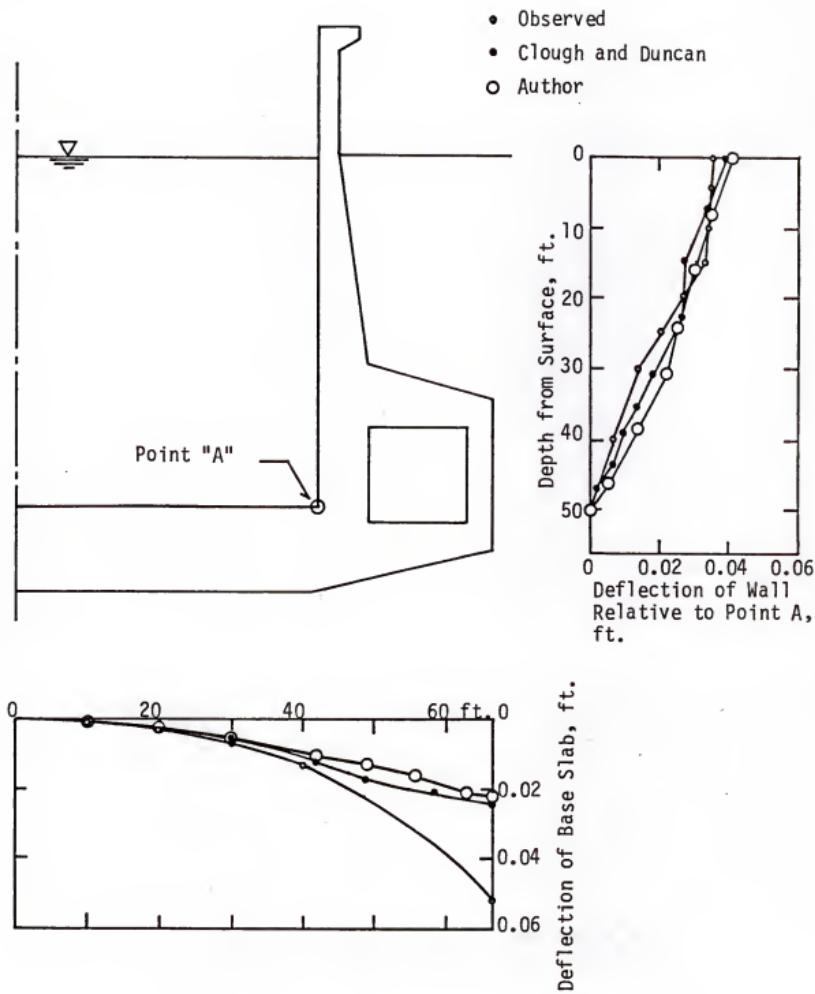


Figure 9-7 Structural deflections for case III'

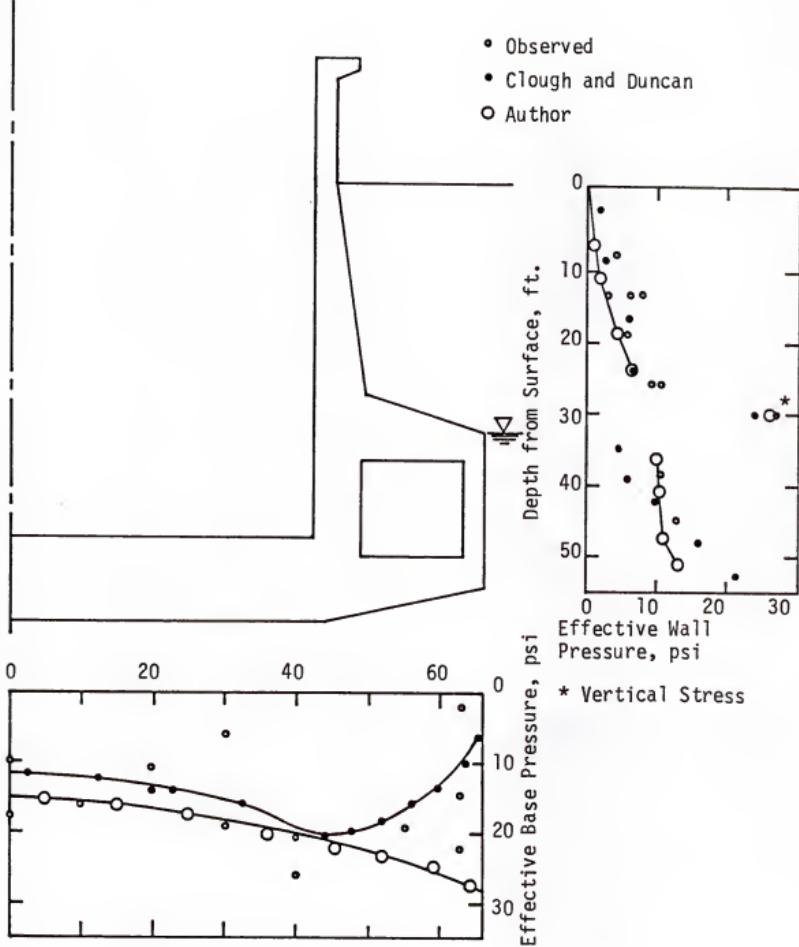


Figure 9-8 Effective earth pressures for case II'

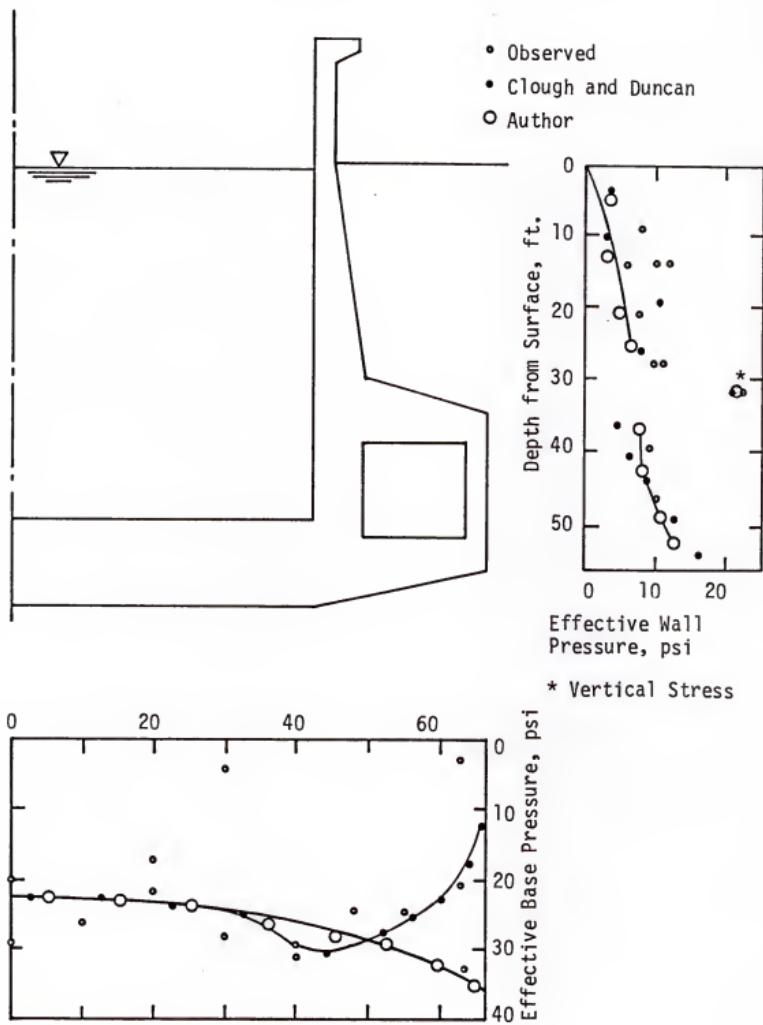


Figure 9-9 Effective earth pressures for case III'

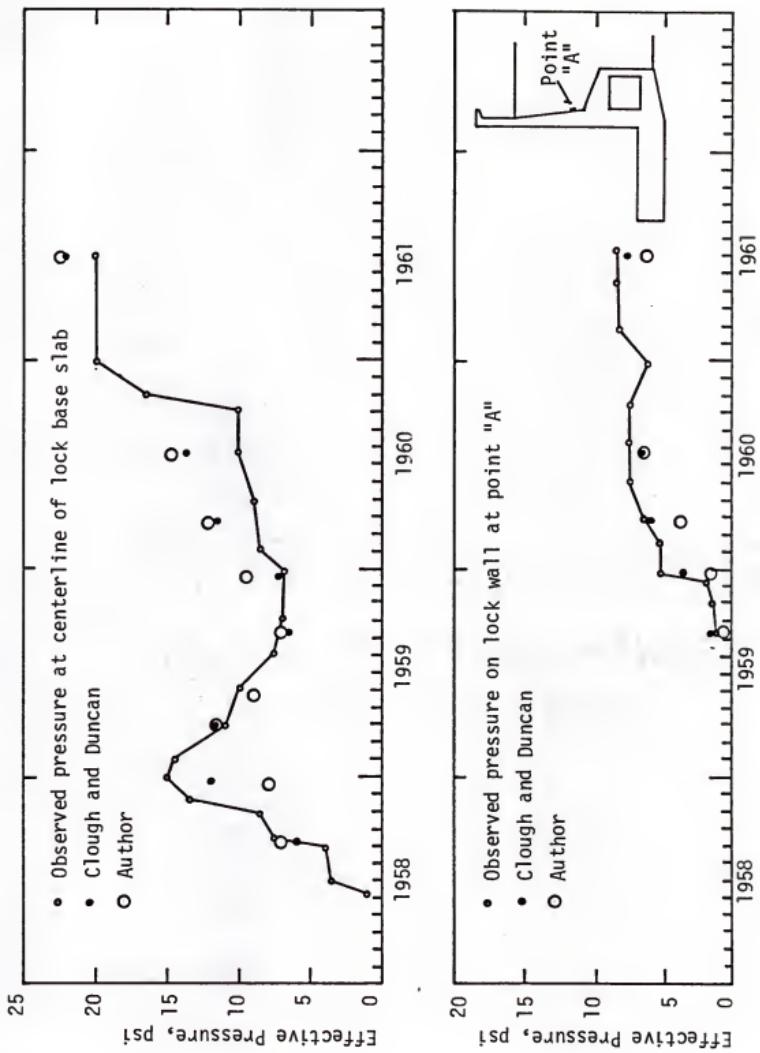


Figure 9-10 Variation of effective earth pressures with time

Braced Excavation in Soft Clay

Introduction

The problem, which is known as Vaterland 1, was a test section on the Oslo subway system which was instrumented and the field data were reported by the Norwegian Geotechnical Institute [16]. A nonlinear material finite element analysis of this problem was reported by Mana [64] and the material parameters given by him are used in this analysis. It should be mentioned here that the material properties were based on limited triaxial tests on the soil. The excavation site and a soil profile is illustrated in Figure 9-11. The results from the present analysis, which considers both material and geometric nonlinearities, are compared with those from Mana and the field observations.

Problem Description

The soil behavior is simulated as elastic-plastic by employing the Drucker-Prager model. Figure 9-12 shows the finite element mesh. Bar elements are used to model the struts and thin-layer elements were used where the soil is in contact with the wall. The material properties used for the analysis are shown in Table 9-2. The wall, strut and interface element materials were assumed to be linear elastic. The analysis involved eight stages which included insitu stress conditions with a number of excavation and strut-installation sequences:

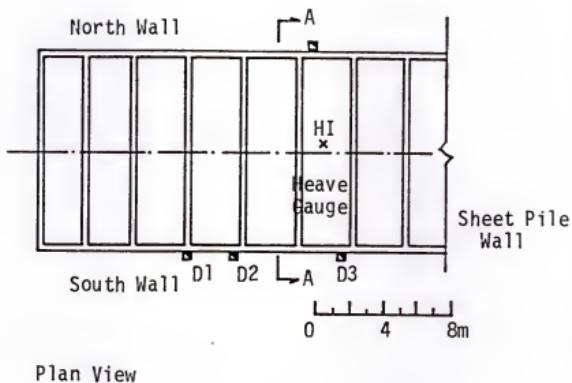
Stage 1: Compute initial stresses, install the wall and excavate to elevation
+0.2 meters.

Stage 2: Install strut A and excavate to elevation -2.0 meters.

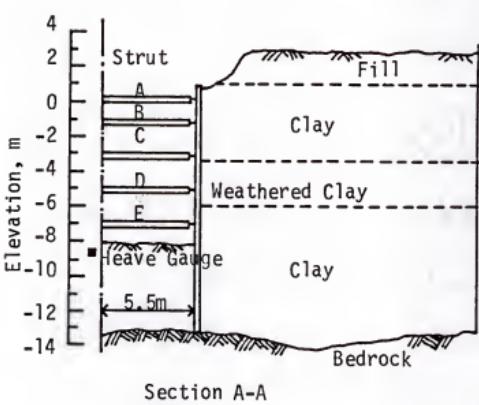
Stage 3: Install strut B and excavate to elevation -3.0 meters.

Stage 4: Install strut C and excavate to elevation -4.0 meters.

Stage 5: Excavate to elevation -5.0 meters.



Plan View



Section A-A

Figure 9-11 Vaterland 1, site and soil profile

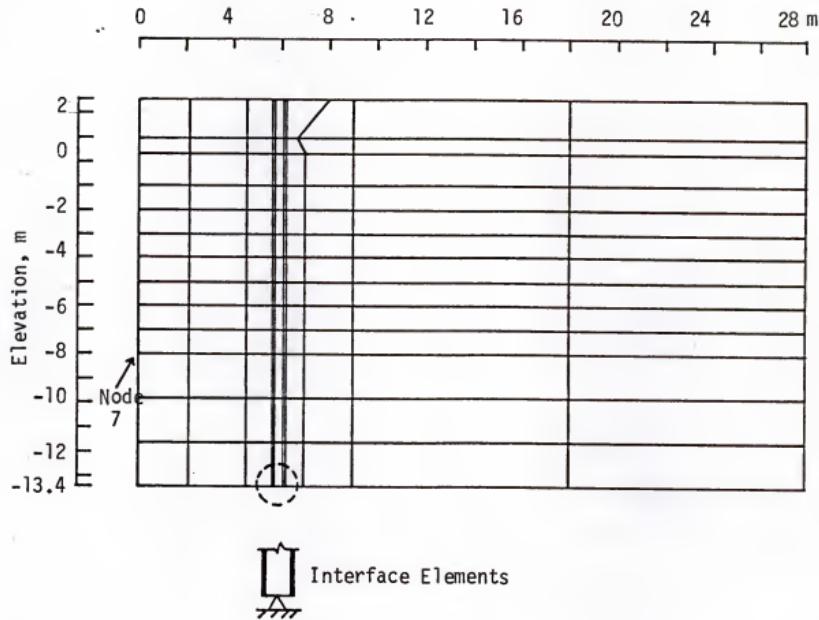


Figure 9-12 Finite element mesh for Vaterland 1 excavation

TABLE 9-2
Soil Properties for Vaterland 1 [64]

Material Number	E (t/m ²)	v	C (t/m ²)	ø	γ (t/m ²)	k _o
1	800.0	0.4933	4.0	0	1.95	0.65
2	760.0	0.4933	3.8	0	1.95	0.65
3	900.0	0.4933	4.5	0	1.95	0.65
4	500.0	0.4967	5.0	0	1.95	0.65
5	410.0	0.4967	4.1	0	1.95	0.65
6	368.0	0.30	1.42	0	1.95	0.65

Wall properties $E = 2.76 \times 10^6 \text{ t/m}^2$, $v = 0.33$

Interface properties $E = 450.0 \text{ t/m}^2$, $v = 0.33$, $G = 0.10 \text{ t/m}^2$

Strut properties Strut A, D Area = 1.0 m²

$$E = 912.0 \text{ t/m}^2$$

Strut B, C, E Area = 1.0 m²

$$E = 4560.0 \text{ t/m}^2$$

Stage 6: Install strut D and excavate to elevation -6.0 meters.

Stage 7: Excavate to elevation -7.0 meters.

Stage 8: Install strut E and excavate to elevation -0.8 meters.

Note that the struts were not prestressed.

Discussion of Results

Soil and sheet pile deformation. Figures 9-13, 9-14, 9-15 and 9-16 show sheet pile deflections and settlements behind the wall for stages 2, 5, 7 and 8 respectively. The sheet pile deflections computed by the present analysis are generally in agreement with the results given by Mana and with field measurements. However, the calculated wall deflections at later stages appear to be less than observed. It should be noted here that the present analysis is performed based on the assumption of undrained conditions. However, since most of the field measurements were taken after a period of time, the obtained results may have been affected by such factors as consolidation and creep. Also, the strength of clay adjacent to the excavated surface may have decreased with time. These could be the reasons for underestimation of the deformation of the sheet pile and of the soil.

The heave of the soil at node 7 is shown in Figure 9-17 for all stages. The present analysis predicts values which are smaller than those from field measurements. This is probably again due to the underestimation of the deformation of the sheet pile.

Strut Load. The axial force in each strut is shown in Figures 9-18 and 9-19. The results obtained using the present analysis show slight improvement compared to the results by Mana. However, for strut D, the obtained results are not in agreement with field measurements. This may be due to unreliable test data because strut D was installed at later stage and it seems unlikely that it will have the axial load much greater than those installed earlier.

Earth Pressure. Figures 9-20, 9-21, 9-22 and 9-23 show the earth pressures acting on the wall for stages 2, 3, 5 and 7, respectively. The values given by the author were obtained from the interface elements which are adjacent to the wall. The predicted values are in good agreement with the field measurements and have the same general trend predicted by Mana.

The correlation between calculated values and field measurements can be influenced by a number of factors, the chief among them being the material model for the soil. The predictions using the Drucker-Prager model are considered to be good overall, and they may be improved using more sophisticated models. However, it is believed that the finite element analysis procedure presented in this study can be used for future studies.

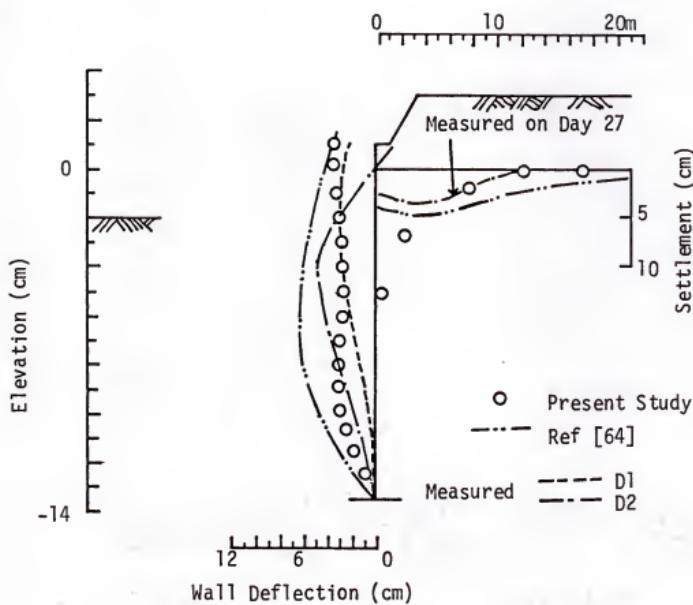


Figure 9-13 Wall and the soil deformation (Stage 2)

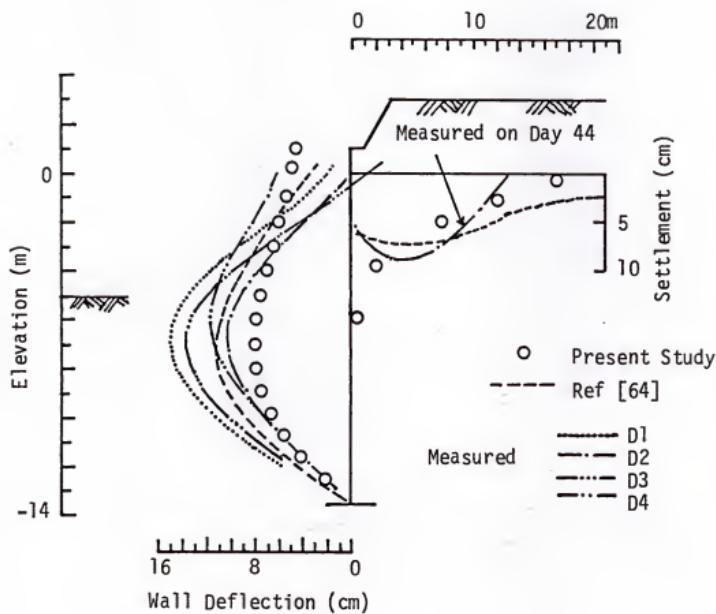


Figure 9-14 Wall and the soil deformation (Stage 5)

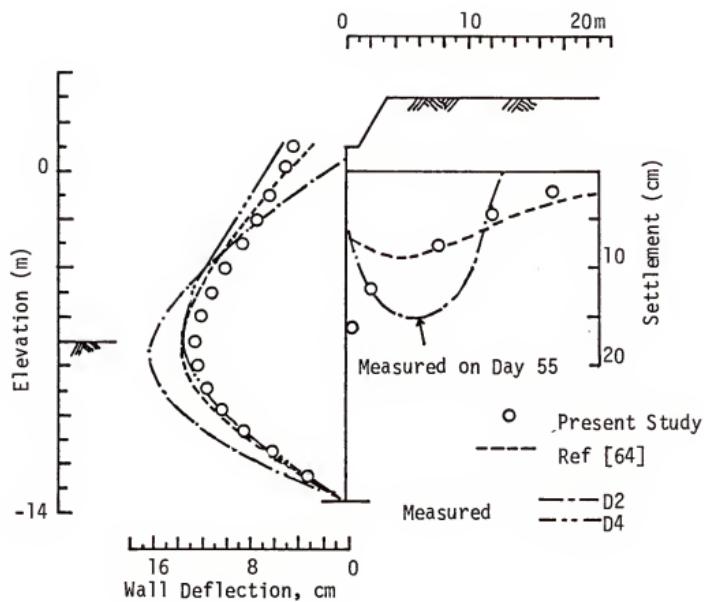


Figure 9-15 Wall and the soil deformation (Stage 7)

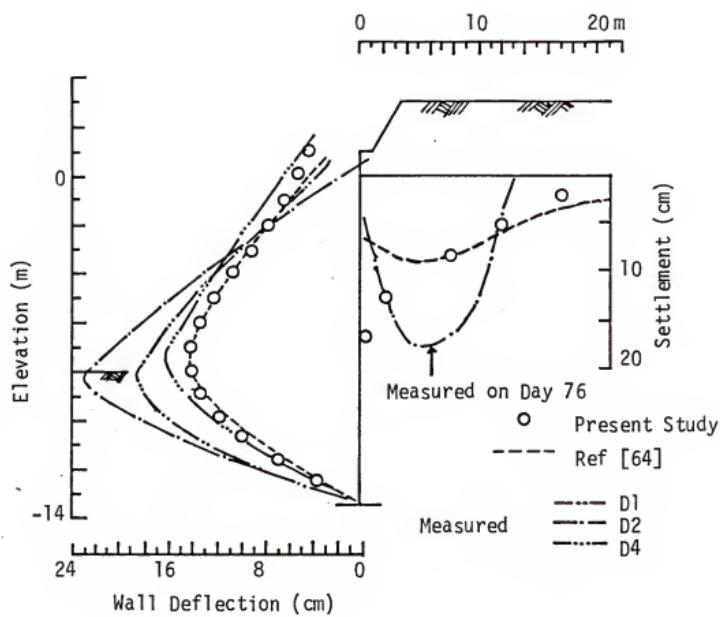


Figure 9-16 Wall and the soil deformation (Stage 8)

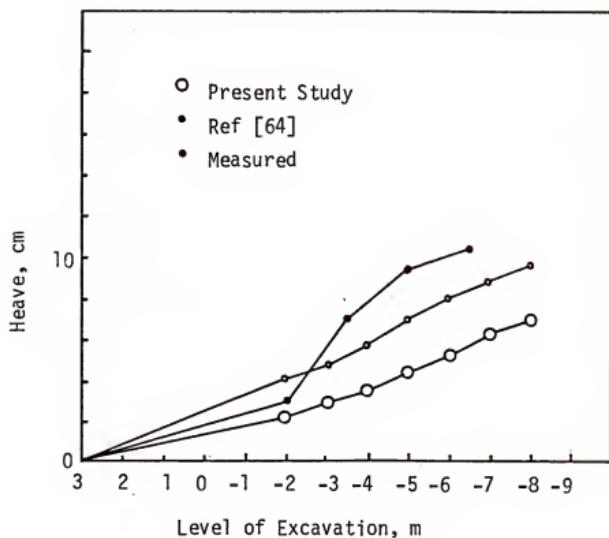
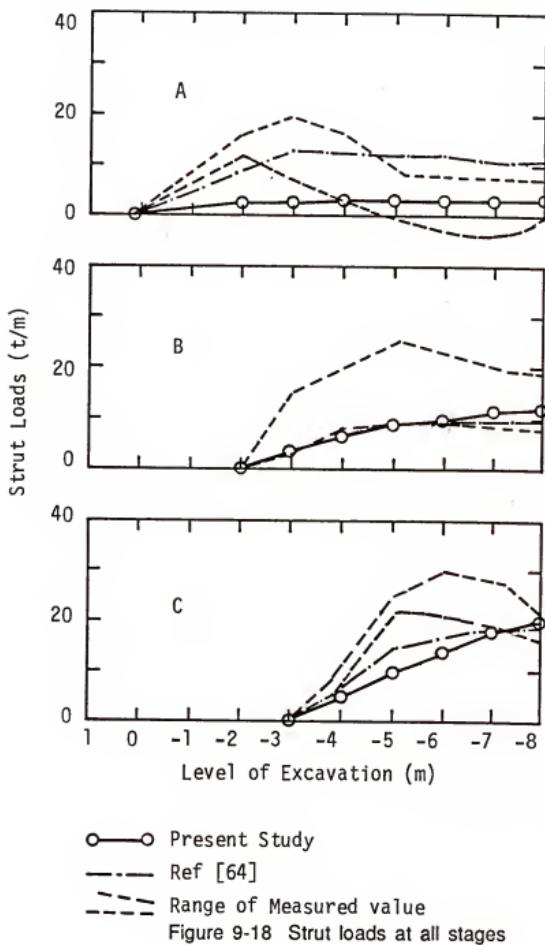


Figure 9-17 Heave at node 7



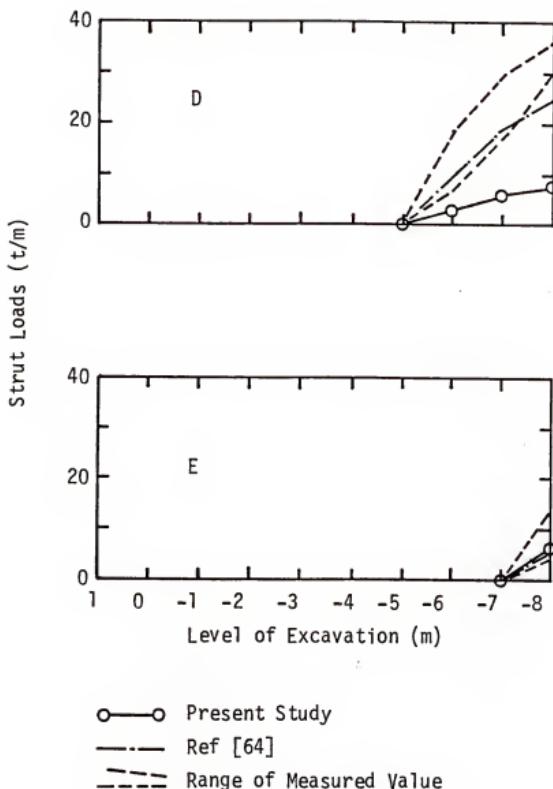


Figure 9-19 Strut loads at all stages

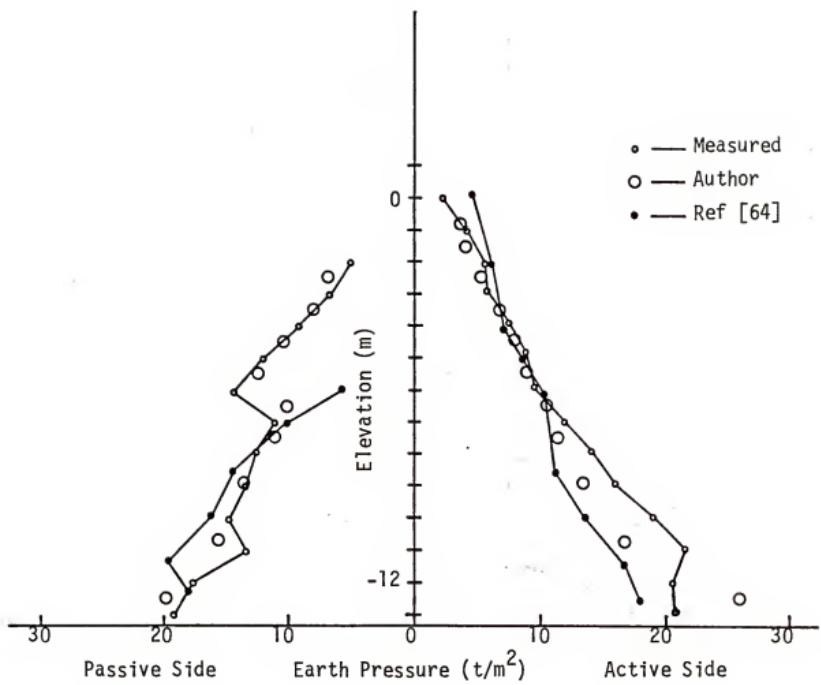


Figure 9-20 Earth pressure (Stage 2)

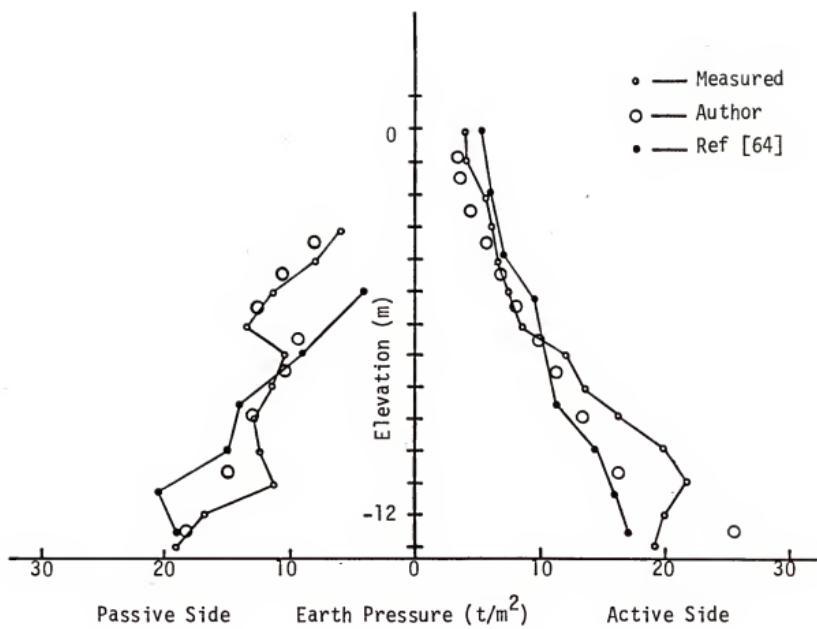


Figure 9-21 Earth pressure (Stage 3)

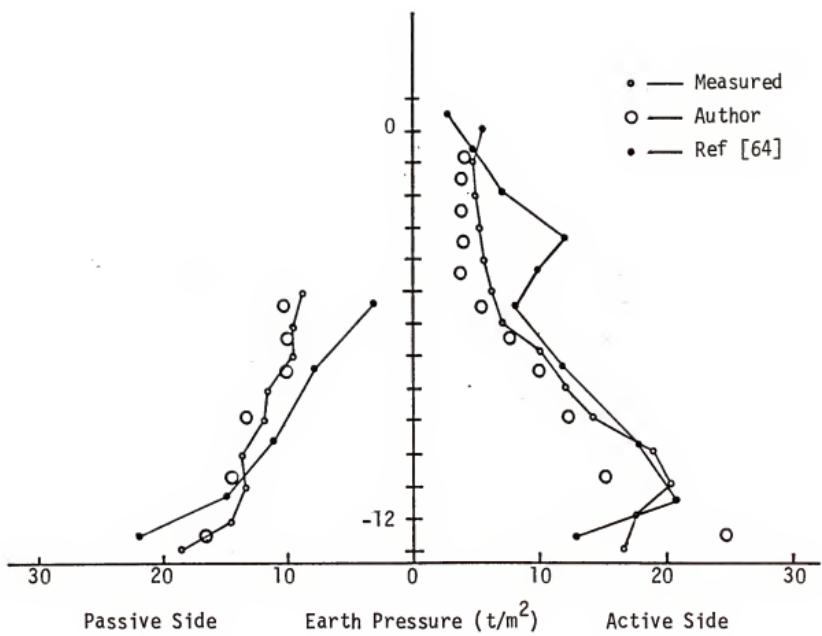


Figure 9-22 Earth pressure (Stage 5)

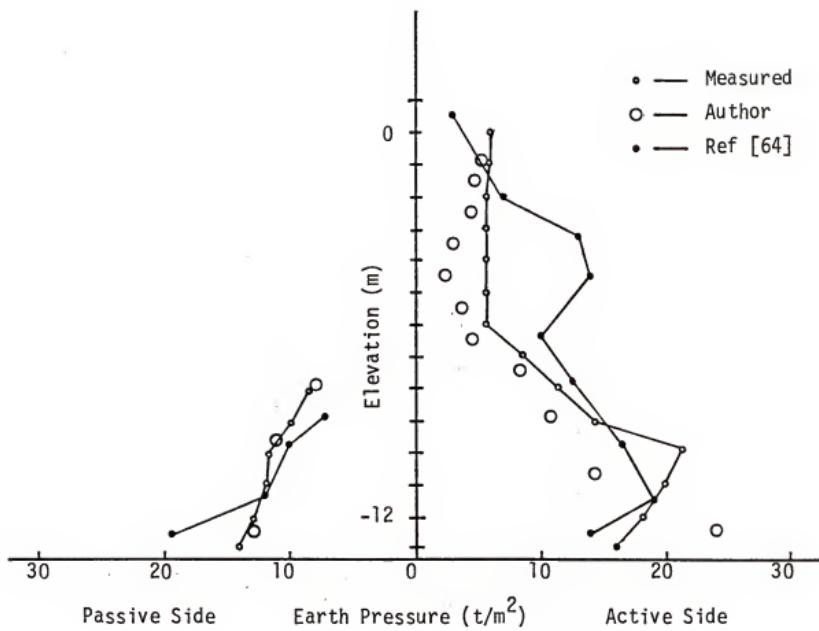


Figure 9-23 Earth pressure (Stage 7)

CHAPTER 10

CONCLUSIONS

The principal result of this research has been the development of a nonlinear geometric and material finite element analytical method for predicting the behavior of the U-frame-soil interaction including construction sequences. The method has been put into a useful form by developing computer program FEMCON.

This chapter first summarizes the work done in this study, and then conclusions of the research are given. Finally, recommendations for future studies are discussed.

Summary

A general incremental finite element formulation for elastic-plastic bodies subjected to large strains and large deformations is presented. The formulation is done in updated Lagrangian coordinates and is further expanded to include models simulating construction sequences. Based on the formulation, a computer program, FEMCON, is developed. The program has sufficient generality to analyze two-dimensional soil-structure interaction problems.

Three plastic constitutive models were implemented. They are: 1) Mohr-Coulomb model, 2) Drucker-Prager model, and 3) Cap model. The tangent stiffness matrices for Mohr-Coulomb and Cap plasticity models were derived in Appendices A and B respectively. A special "thin" interface element which uses a linear elastic or hyperbolic constitutive model was used to model the interface between soil and structure.

Models for simulating insitu stresses, dewatering, excavation and embankment are presented, but greater emphasis is given to excavation. The excavation model used earlier in the displacement finite element method was unable to predict the accurate

stress distribution on an excavated surface. A new model utilizing the stresses at Gauss (integration) points has been proposed and verified. A one-dimensional bar element which can be installed or removed was also included for modeling tie-backs and other reinforcement such as struts, anchors and braces. Unloading is generally associated with construction sequences. To solve nonlinear problems with unloading, difficulties may arise when the tangent stiffness method is used. A mixed tangent stiffness method which can detect unloading and converge as fast as the tangent stiffness method was proposed.

An attempt was made to use the discrete element method to solve the U-frame structures using the soil response curves generated from the finite element program FEMCON. It is found that these curves need to be adjusted in order that the discrete element solution to be compatible with the finite element solution. It is also found that, to obtain finite element solutions to problems solved by the limit equilibrium approach, new plasticity interface models are required.

A variety of soil mechanics problems was solved to verify the formulation and computer program. Two field problems were analyzed to evaluate the formulation and program with respect to practical problems.

Conclusions

Based on the numerical predictions of a variety of typical soil mechanics problems, which showed satisfactory comparisons with results from other numerical schemes and experimental observations, it is felt that the finite element procedure and program presented herein can provide a general solution scheme for the two-dimensional soil-structure interaction problems.

Due to the broad analytical scope of the work, conclusions about nonlinear behavior for U-frame structures in elastic-plastic medium will not be drawn at this time. Before such conclusions are drawn, the available and newly developed constitutive

models should be incorporated into the computer program and then more detailed studies should be made using the computer program in conjunction with existing experimental data and as an aid to planned experimental tests.

Use of the finite element method in geomechanics problems is not new, but this study is one of the first to incorporate nonlinear material behavior, nonlinear geometric effect, soil-structure interaction and simulation of construction sequences into one efficient finite element program. It is believed that since the real soil-structure problems generally are constructed in sequences and soil is highly associated with nonlinear material and geometric behavior, the method presented herein will have a high potential for continued research and application.

Recommendations for Future Study

The program was intended to be capable of simulating real soil-interaction problems including construction sequences. There are, however, some modifications that could be made in order to make the program more convenient and powerful to the user. Several possible modifications are

1. Add a restart option. The analysis of a particular problem may be discontinued at any prescribed points and then resumed in a later analysis through the use of the restart option.
2. Add more constitutive soil models. Not all soil behaviors can be represented by the soil models implemented.
3. Add more one- and two-dimensional elements in the element library for much more flexible domain discretization.
4. Add interactive computer graphics to speed up the time-consuming process of accurate data preparation and output interpretation.

Although computer program FEMCON can only solve two-dimensional problems at present, it has the capability of being expanded to include three-dimensional problems

by simply adding three-dimensional elements and using three-dimensional constitutive models.

Finally, the scope of this study was limited to the time-independent static elastic-plastic analysis of structures in soil medium. However, many soil-structure interaction problems have important time-varying creep and time-dependent dynamic characteristics and sometimes may be required to consider the steady state and transient state seepage. The work done in this research could serve as a guide for efforts to extend the nonlinear analysis technique to include these effects.

REFERENCES

1. Baladi, G. Y. and B. Rohzani, "Elastic-Plastic Model for Saturated Sand," Journal of the Geotechnical Engineering Division, ASCE, Vol. 105, No. GT4, April 1979, p. 465-481.
2. Banerjee, P. K. and R. Butterfield, Boundary Element Methods in Engineering Science, McGraw-Hill, London, 1981.
3. Bathe, K. J., ADINA – Static and Dynamic Geometric and Material Nonlinear Analysis, Report No. 82448-2, M.I.T., Cambridge, MA, 1976.
4. Bathe, K. J., Finite Element Procedures in Engineering Analysis, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.
5. Bathe, K. J. and S. Bolourchi, "A Geometric and Material Nonlinear Plate and Shell Element," Journal of Computers and Structures, Vol. 11, 1980, p. 23-48.
6. Bathe, K. J., E. Ramm and E. L. Wilson, "Finite Element Formulations for Large Deformation Dynamic Analysis," International Journal for Numerical methods in Engineering, Vol. 9, 1975, p. 353-386.
7. Beer, G., "An Isoparametric Joint/Interface Element for Finite Element Analysis," International Journal for Numerical Methods in Engineering, Vol. 21, 1985, p. 585-600.
8. Bishop, A. W., "Test Requirements for Measuring the Coefficient of Earth Pressure at Rest," Proceedings, Conference on Earth Pressure Problems, Vol. 1, Brussels, 1958, p. 2-14.
9. Chen, W. F., Limit Analysis and Soil Plasticity, Elsevier Scientific Publishing Company, Amsterdam, 1975.
10. Chen, W. F., Plasticity in Reinforced Concrete, McGraw-Hill, New York, 1982.
11. Chen, W. F. and G. Y. Baladi, Soil Plasticity – Theory and Implementation, Elsevier Science Publishing Company, Amsterdam, 1985.
12. Chen, W. F. and A. F. Saleeb, Constitutive Equations for Engineering Materials, Vol. 1, John Wiley and Son, New York, 1982.
13. Chen, W. F. and A. F. Saleeb, Constitutive Equations for Engineering Materials, Vol. 2 , John Wiley and Son, New York, 1986.
14. Christian, J. T. and I. H. Wong, "Errors in Simulating Excavation in Elastic Media by Finite Elements," Soils and Foundations, Japanese Society of Soil Mechanics and Foundation Engineering, Vol. 13, No. 1, March 1973, p. 1-10.

15. Chandrasekaran, V. S. and G. J. King, "Simulation of Excavation Using Finite Elements," Journal of the Geotechnical Engineering Division, ASCE, Vol. 100, No. GT9, September 1974., p. 1086-1089.
16. Clausen, C. J. F., Finite Element Analysis of Strutted Excavation at Vaterland 1, Report No. 52601-2, Norwegian Geotechnical Inst., Oslo, Norway, 1971.
17. Clough G. W. and J. M. Duncan, Finite Element Analyses of Port Allen and Old River Locks, Report No. TE 69-3, College of Engineering, Office of Research Services, University of California, Berkeley, California, September 1969.
18. Clough, G. W. and J. M. Duncan, Finite Element Analyses of Retaining Wall Behavior, Journal of the Soil Mechanics and Foundations Division, ASCE, Vol. 97, No. SM12, December 1970, p. 1657-1673.
19. Cook, R. D., Concepts and Applications of Finite Element Analysis, 2nd Edition, John Wiley and Son, New York, 1981.
20. Dafalias, Y. F., "Corotational Rates for Kinematic Hardening at Large Plastic Deformation," Transactions of the ASME, Journal of Applied Mechanics, Vol. 50, September 1983, p. 561-565.
21. Davidson, H. L. and W. F. Chen, Elastic-Plastic Large Deformation Response of Clay to Footing Loads, Report No. 355-18, Fritz Engineering Laboratory, Lehigh University, 1974.
22. Desai, C. S. and J. F. Abel, Introduction to the Finite Element Method, Von Nostrand Reinhold, New York, 1972.
23. Desai, C. S. and J. Christian, Editors, Numerical Methods in Geotechnical Engineering, McGraw-Hill, New York, 1977.
24. Desai, C. S. and J. G. Lightner, "Mixed Finite Element Procedure for Soil-Structure Interaction and Construction Sequences," International Journal for Numerical Methods in Engineering, Vol. 21, 1985, p. 801-824.
25. Desai, C. S. and H. V. Phan, "Three-Dimensional Finite Element Analysis Including Material and Geometric Nonlinearities," in Computational Methods in Nonlinear Mechanics, J. T. Oden, Ed., North-Holland Publishing Company, Amsterdam, 1980.
26. Desai, C. S., H. V. Phan and J. V. Perumpral, "Mechanics of Three-Dimensional Soil-Structure Interaction," Journal of the Engineering Mechanics Division, ASCE, Vol. 108, No. EM5, October 1982, p. 731-747.
27. Desai, C. S. and S. M. Sargand, "Hybrid FE Procedure for Soil-Structure Interaction," Journal of the Geotechnical Engineering Division, ASCE, Vol. 110, No. 4, April 1984, p. 473-486.
28. Desai, C. S. and H. J. Siriwardane, Constitutive Laws for Engineering Materials with Emphasis on Geologic Materials, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1984.

29. Desai, C. S., M. M. Zaman, J. G. Lightner and H. J. Siriwardane, "Thin-Layer Element for Interfaces and Joints," International Journal for Numerical and Analytical Methods in Geomechanics, Vol. 8, No. 1, 1984, p.19-43.
30. Dhatt, G. and G. Touzot, The Finite Element Method Displayed, English Translation by G. Cantin, John Wiley and Son, New York, 1984.
31. Dimaggio, F. L. and I. S. Sandler, "Material Model for Granular Soils," Journal of the Engineering Mechanics Division, ASCE, Vol. 97, No. EM3, 1971, p. 935-950.
32. Drucker, D. C., R. E. Gibson and D. J. Hankel, "Soil Mechanics and Work Hardening Theories of Plasticity," Transactions of the ASCE, Vol. 122, Paper No. 2864, 1957, p. 338-346.
33. Drucker, D. C. and W. Prager, "Soil Mechanics and Plasticity Analysis or Limit Design," Quarterly of Applied Mathematics, Vol. 10, No. 2, 1952, p. 157-165.
34. Duncan, J. M., P. Byrne, K. S. Wong and P. Mabry, Strength, Stress-Strain and Bulk Modulus Parameters for Finite Element Analyses of Stresses and Movements in Soil Masses, Report No. UCB/GT/78-2, University of California, Berkeley, California, April 1978.
35. Duncan, J. M. and C. Y. Chang, "Nonlinear Analysis of Stress and Strain in Soils," Journal of the Soil Mechanics and Foundations Division, ASCE, Vol. 96, No. SMS, September 1970, p. 1629-1653.
36. Duncan, J. M. and G. W. Clough, "Finite Element Analyses of Port Allen Lock," Journal of the Soil Mechanics and Foundations Division, ASCE, Vol. 97, No. SM8, August 1971, p. 1053-1068.
37. Fang, Y. S. and I. Ishibashi, "Static Earth Pressures with Various Wall Movements," Journal of the Geotechnical Engineering, ASCE, Vol. 112, No. 3, March 1986, p. 317-333.
38. Frederick, D. and T. S. Chang, Continuum Mechanics, Scientific Publishers, Inc., Boston, 1972.
39. Fung, Y. C., Foundations of Solid Mechanics, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1965.
40. Ghaboussi, J., E. L. Wilson and J. Isenberg, "Finite Element for Rock Joints and Interfaces," Journal of the Soil Mechanics and Foundation Engineering, ASCE, Vol. 99, SM10, October 1973, p. 833-848.
41. Goodman, R. E. and C. B. Brown, "Dead Load Stresses and the Instability of Slopes," Journal of the Soil Mechanics and Foundation Division, ASCE, Vol. 89, No. SM3, May 1963, p. 103-134.
42. Goodman, R. E., R. L. Taylor and T. L. Brekke, "A Model for the Mechanics of Jointed Rock," Journal of the Soil Mechanics and Foundations Division, ASCE, Vol. 94, No. SM5, May 1968, p. 637-659.

43. Goodman, R. E. and J. Christopher, "Finite Element Analysis for Discontinuous Rocks," in Numerical Methods in Geotechnical Engineering, C. S. Desai and J. T. Christian, Editors, McGraw-Hill, New York, 1977, p.148-175.
44. Haliburton, T. A., "Numerical Analysis of Flexible Retaining Structures," Journal of the Soil Mechanics and Foundations Division, ASCE, Vol. 94, No. SM6, November 1968, p. 1233-1251.
45. Haliburton, T. A., Soil Structure Interaction – Numerical Analysis of Beams and Beam-Columns, Technical Publication No. 14, School of Civil Engineering, Oklahoma State University, 1971.
46. Hays, C. O., A Nonlinear Analysis of Statically Loaded Plane Frames Using A Discrete Element Model, Ph.D. Dissertation, University of Texas at Austin, 1971.
47. Hays, C. O. and H. Matlock, "Nonlinear Discrete Element Analysis of Frames," Journal of the Structural Division, ASCE, Vol. 99, No. ST10, October 1973, p. 2011-2030.
48. Herrman, L. R., "Finite Element Analysis of Contact Problems," Journal of the Engineering Mechanics Division, ASCE, Vol. 104, No. EM5, October 1978, p. 1043-1057.
49. Hibbit, H. D., P. V. Marcal and J. R. Rice, "A Finite Element Formulation for Problems of Large Strain and Large Displacement," International Journal of Solids and Structures, Vol. 6, 1970, p. 1069-1086.
50. Hill, R., The Mathematical Theory of Plasticity, Clarendon Press, Oxford, England, 1950.
51. Holubec, I., "Elastic Behavior of Cohesionless Soil," Journal of the Soil Mechanics and Foundations Division, ASCE, Vol. 94, No. SM6, November 1968, p. 1215-1231.
52. Huang, Y. P., Nonlinear, Incremental, 2-D and 3-D Finite Element Analysis of Geotechnical Structures Using Interactive Computer Graphics, Ph.D. Dissertation, Cornell University, August 1983.
53. Huebner, K. H. and E. A. Thornton, The Finite Element Method for Engineers, John Wiley and Son, New York, 1982.
54. Ishizaki, T. and K. J. Bathe, "On Finite Element Large Displacement and Elastic-Plastic Dynamic Analysis of Shell Structures," Journal of Computers and Structures, Vol. 12, 1980, p. 309-318.
55. Kioussis, P. D., G. Z. Voyatzis and M. T. Tumay, "A Large Strain Theory for the Two Dimensional Problems in Geomechanics," International Journal for Numerical and Analytical Methods in Geomechanics, Vol. 10, 1986, p. 17-39.
56. Kondner, R. L., "Hyperbolic Stress-Strain Response: Cohesive Soils," Journals of the Soil Mechanics and Foundation Division, ASCE, Vol. 94, No. SM1, February 1963, p. 115-143.

57. Kulhawy, F. H., J. M. Duncan and H. B. Seed, Finite Element Analyses of Stresses and Movements in Embankments During Construction, Geotechnical Engineering Report No. TE 69-4, Department of Civil Engineering, University of California, Berkeley, California, 1969.
58. Lade, P. V., "Elastic-Plastic Stress-Strain Theory for Cohesionless Soil with Curved Yield Surfaces," International Journal of Solids and Structures, Vol. 13, 1977, p. 1019-1035.
59. Lade, P. V. and J. M. Duncan, "Elastoplastic Stress-Strain Theory for Cohesionless Soil," Journal of the Geotechnical Engineering Division, ASCE, Vol. 101, No. GT10, October 1975, p. 1037-1053.
60. Lee, E. H., R. L. Mallett and T. B. Wertheimer, "Stress Analysis for Anisotropic Hardening in Finite-Deformation Plasticity," Transactions of the ASME, Journal of Applied Mechanics, Vol. 50, September 1983, p. 554-560.
61. Lightner, III, J. G., A Mixed Finite Element Procedure for Soil-Structure Interaction Including Construction Sequences, Ph.D. Dissertation, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, June 1981.
62. Lightner, III, J. G., Improved Numerical Procedures for Soil-Structure Interaction Including Simulation of Construction Sequences, M.S. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1979.
63. Malvern, L. E., Introduction to the Mechanics of a Continuous Medium, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1969.
64. Mana, A. I., Finite Element Analyses of Deep Excavation Behavior in Soft Clay, Ph.D. Dissertation, Stanford University, March 1978.
65. Makhlof, H. M. and J. J. Stewart, "Factors Influencing the Modulus of Elasticity of Dry Sand," Proceedings of the 6th International Conference on Soil Mechanics and Foundations Engineering, Vol. 1, Montreal, 1965, p. 298-302.
66. Mayne, P. M. and F. H. Kulhawy, "Ko-OCR Relationships in Soil," Journal of the Geotechnical Engineering Division, ASCE, Vol. 108, No. GT6, June 1982, p. 851-872.
67. Meldelson, A., Plasticity: Theory and Application, Macmillan, New York, 1968.
68. Mizuno, E., Plasticity Modeling of Soils and Finite Element Applications, Ph.D. Thesis, School of Civil Engineering, Purdue University, West Lafayette, Ind., 1981.
69. Mizuno, E. and W. F. Chen, "Analysis of Soil Response with Different Plasticity Models," in Limit Equilibrium, Plasticity and Generalized Stress-Strain Application in Geotechnical Engineering, R. N. Yong and E. T. Selig, Editors, ASCE, New York, 1980, p. 115-138.
70. Murray, P. W. and E. L. Wilson, "Finite Element Large Deflection Analysis of Plates," Journal of the Engineering Mechanics Division, ASCE, Vol. 94, 1965, p. 143-155.

71. Nagtegaal, J. C., D. M. Park and J. R. Rice, "On Numerically Accurate Finite Element Solutions in the Fully Plastic Range," Computer Methods in Applied Mechanics and Engineering, Vol. 4, 1974, p. 113-135.
72. Nayak, G. C. and O. C. Zienkiewicz, "A Convenient Form of Invariants and Its Application in Plasticity," Proc. ASCE, Vol. 98, No. ST4, 1972, p. 949-954.
73. Osaimi, A. E. and W. G. Clough, "Pore-Pressure Dissipation During Excavation," Journal of the Geotechnical Engineering Division, ASCE, Vol. 5, No. GT4, April 1979, p. 481-498.
74. Phan, H. V., C. S. Desai, S. Sture and J. Perumpral, "Three-Dimensional Geometric and Material Nonlinearities Analysis of Some Problems in Geomechanics," in Proc. 3rd International Conference Numerical Methods in Geomechanics, Aachen, F.R. Germany, 1979, p. 67-76.
75. Radhakrishnan, N. and H. W. Jones, "Documentation for Modified UFRAME Program," U. S. Army Engineer Waterways Experiment Station, Vicksburg, Mississippi, 1976.
76. Ramm, E., "A Plate/Shell Element for Large Deflections and Rotations," in Formulations and Computational Algorithms in Finite Element Analysis, K. J. Bathe, J. T. Oden and W. Wunderlich, Editors, M.I.T. Press, Cambridge, Mass., 1977. p. 264-291.
77. Rao, S. S., The Finite Element Method in Engineering, Pergamon Press, Oxford, England, 1982.
78. Reismann, H. and P. S. Pawlik, Elasticity – Theory and Application, John Wiley and Son, New York, 1980.
79. Sandler, I. S., F. L. Dimaggio and G. Y. Baladi, "Generalized Cap Model for Geological Materials," Journal of the Geotechnical Division, ASCE, Vol. 102, No. Em2, July 1976, p. 683-699.
80. Sandler, I. S. and D. Rubin, "An Algorithm and a Modular Subroutine for the Cap Model," International Journal for Numerical and Analytical Methods in Geomechanics, Vol. 3, 1979, p. 173-186.
81. Sargand, S. M., A Hybrid Finite Element Procedure for Soil-Structure Interaction Including Construction Sequences, Ph.D. Dissertation, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, June 1981.
82. Schofield, A. and P. Wroth, Critical State Soil Mechanics, McGraw-Hill, New York, 1968.
83. Scott, R. F., "Plasticity and Constitutive Relations in Soil Mechanics," Journal of Geotechnical Engineering, ASCE, Vol. 111, No. 5, May 1985, p. 563-605.
84. Sheriff, M. A., I. Ishibashi and C. D. Lee, "Earth Pressures Against Rigid Retaining Walls," Journal of the Geotechnical Division, ASCE, Vol. 108, No. GT5, May 1982, p. 679-695.

85. Sheriff, M. A., Y. S. Fang and R. I. Sheriff, "K_a and K_o Behind Rotating and Non-Yielding Walls," Journal of the Geotechnical Division, ASCE, Vol. 110, No. 1, January 1984, p. 41-56.
86. Sherman, W. C. and C. C. Trahan, Analysis of Data from Instrumentation Program, Port Allen Lock, Technical Report S-68-7, U. S. Army Engineer Waterways Experiment Station, Corps of Engineers, Vicksburg, Mississippi, September 1968.
87. Siriwardane, H. J., Nonlinear Soil-Structure Interaction Analysis of One-, Two-, and Three-Dimensional Problems Using Finite Element Method, Ph.D Dissertation, VPI & SU, Blacksburg, Virginia, November 1980.
88. Smith, I. M., Programming the Finite Element Method with Application to Geomechanics, John Wiley and Son, Chichester, England, 1982.
89. Sture, S., C. S. Desai and R. Janardhanam, "Development of a Constitutive Law for an Artificial Soil," in Proc. 3rd International Conference Numerical Methods in Geomechanics, Aachen, F.R. Germany, 1979, p. 309-317.
90. Taylor, L. M. and E. B. Becker, "Some Computational Aspects of Large Deformation, Rate-Dependent Plasticity Problems," Computer Methods in Applied Mechanics and Engineering, Vol. 41, 1983, p. 251-277.
91. Terzaghi, K., Theoretical Soil mechanics, John Wiley and Son, New York, 1943.
92. Tillerson, J. R., J. A. Stricklin and W. E. Hairler, "Numerical Methods for the Solution of Nonlinear Problems in Structural Analysis," in Numerical Solution of Nonlinear Structural Problems, R. F. Hartung, Ed., Applied Mechanics Division, Vol. 6, ASME, New York, 1973, p. 67-101.
93. Timoshenko, S. P. and J. N. Goodier, Theory of Elasticity, Third Edition, McGraw-Hill, New York, 1970.
94. Washizu, K., Variational Methods in Elasticity and Plasticity, Pergamon Press, Oxford, England, 1975.
95. Yang, H. C., FEMCON – User's Guide, Department of Civil Engineering, University of Florida, 1988.
96. Young, R. N. and H. Y. Ko, Editors, Limit Equilibrium, Plasticity and Generalized Stress-Strain in Geotechnical Engineering," ASCE, New York, 1981.
97. Zienkiewicz, O. C., The Finite Element Method, McGraw-Hill, London, England, 1977.
98. Zienkiewicz, O. C., Valliapan, S. and King, I. P., "Elasto-Plastic Solution of Engineering Problems – 'Initial Stress' Finite Element Method," International Journal for Numerical Methods in Engineering, Vol. 1, No.1, 1969, p. 75-100.

APPENDIX A
MOHR-COULOMB ELASTIC-PLASTIC CONSTITUTIVE MATRIX

This appendix derives implicitly the elastic-plastic constitutive matrix for the Mohr-Coulomb model. The matrix is obtained from equation 4.26, 4.29 and 4.30 by proper mathematical manipulation.

$$C^{EP}{}_{ijrs} = C_{ijrs} - \frac{C_{ijkl} \frac{\partial F}{\partial \sigma_{kl}} \frac{\partial F}{\partial \sigma_{mn}} C_{mnr}}{\frac{\partial F}{\partial \sigma_{mn}} C_{mn} u v \frac{\partial F}{\partial \sigma_{uv}} - \frac{\partial F}{\partial k} \frac{\partial k}{\partial \epsilon P_{mn}} \frac{\partial F}{\partial \sigma_{mn}}} \quad (4.26)$$

$$F = \frac{1}{3} I_1 \sin \theta + \sqrt{J_2} (\cos \theta - \frac{\sin \theta \sin \phi}{\sqrt{3}}) - c \cos \phi = 0 \quad (4.29)$$

$$\theta = - \arcsin(-\frac{3\sqrt{3}}{2} \frac{J_3}{J_2^{3/2}}) \quad (4.30)$$

For the plane-strain case, equation 4.26 can be expressed in matrix form as

$$[C^{EP}]_{4x4} = [C]_{4x4} - \frac{[C]_{4x4} \{DF\}_{4x1} \{DF\}_{4x1}^T}{\{DF\}_{1x4}^T [C]_{4x4} \{DF\}_{4x1}} \quad (A.1)$$

where

$$[C] = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & 0 & \nu \\ \nu & 1-\nu & 0 & \nu \\ 0 & 0 & \frac{1-2\nu}{2} & 0 \\ \nu & \nu & 0 & 1-\nu \end{bmatrix} \quad (A.2)$$

$$\{\Delta F\}_{1 \times 4}^T = \left\{ \frac{\partial F}{\partial \sigma_{11}}, \frac{\partial F}{\partial \sigma_{22}}, \frac{\partial F}{\partial \sigma_{12}}, \frac{\partial F}{\partial \sigma_{33}} \right\} \quad (A.3)$$

The partials in equation A.3 are found by differentiating equation 4.29.

$$\frac{\partial F}{\partial \sigma_{11}} = \frac{\partial F}{\partial I_1} \frac{\partial I_1}{\partial \sigma_{11}} + \frac{\partial F}{\partial J_2} \frac{\partial J_2}{\partial \sigma_{11}} + \frac{\partial F}{\partial J_3} \frac{\partial J_3}{\partial \sigma_{11}} \quad (A.4)$$

$$\frac{\partial F}{\partial \sigma_{22}} = \frac{\partial F}{\partial I_1} \frac{\partial I_1}{\partial \sigma_{22}} + \frac{\partial F}{\partial J_2} \frac{\partial J_2}{\partial \sigma_{22}} + \frac{\partial F}{\partial J_3} \frac{\partial J_3}{\partial \sigma_{22}} \quad (A.5)$$

$$\frac{\partial F}{\partial \sigma_{12}} = \frac{\partial F}{\partial I_1} \frac{\partial I_1}{\partial \sigma_{12}} + \frac{\partial F}{\partial J_2} \frac{\partial J_2}{\partial \sigma_{12}} + \frac{\partial F}{\partial J_3} \frac{\partial J_3}{\partial \sigma_{12}} \quad (A.6)$$

$$\frac{\partial F}{\partial \sigma_{33}} = \frac{\partial F}{\partial I_1} \frac{\partial I_1}{\partial \sigma_{33}} + \frac{\partial F}{\partial J_2} \frac{\partial J_2}{\partial \sigma_{33}} + \frac{\partial F}{\partial J_3} \frac{\partial J_3}{\partial \sigma_{33}} \quad (A.7)$$

where

$$\frac{\partial F}{\partial I_1} = \frac{1}{3} \sin \theta \quad (A.8)$$

$$\begin{aligned} \frac{\partial F}{\partial J_2} &= \frac{\partial(\sqrt{J_2} \cos \theta)}{\partial J_2} + \frac{1}{\sqrt{3}} \frac{\partial(\sqrt{J_2} \sin \theta \sin \theta)}{J_2} \\ &= \frac{1}{2\sqrt{J_2}} \cos \theta + \sqrt{J_2} \frac{\partial \cos \theta}{\partial J_2} + \frac{\sin \theta \sin \theta}{2\sqrt{3} \sqrt{J_2}} + \frac{1}{\sqrt{3}} \frac{\sqrt{J_2} \partial(\sin \theta \sin \theta)}{\partial J_2} \\ &= \frac{3 \cos \theta}{2\sqrt{3} \sqrt{3J_2}} [(1 + \frac{\sin \theta \sin 3\theta}{\cos \theta \cos 3\theta} + \frac{\sin 3\theta}{\sqrt{3}} (\frac{\sin \theta}{\cos 3\theta} - \frac{\sin \theta}{\cos \theta})] \end{aligned} \quad (A.9)$$

$$\frac{\partial F}{\partial J_3} = \frac{\partial F}{\partial \theta} \frac{\partial \theta}{\partial J_3}$$

$$\begin{aligned}
 &= \frac{\sqrt{J_2}}{\sqrt{3}} (\sqrt{3} \sin\theta + \cos\theta \sin\phi) \times \frac{-1}{3\cos 3\theta} \frac{-3\sqrt{3}}{2J_2^{3/2}} \\
 &= 1.5 \frac{(\sqrt{3} \sin\theta + \cos\theta \sin\phi)}{\sqrt{3}J_2 \sqrt{3}J_2 \cos 3\theta}
 \end{aligned} \tag{A.10}$$

$$\frac{\partial I_1}{\partial \sigma_{11}} = \frac{\partial I_1}{\partial \sigma_{22}} = \frac{\partial I_1}{\partial \sigma_{33}} = 1 \tag{A.11}$$

$$\frac{\partial I_1}{\partial \sigma_{12}} = 0 \tag{A.12}$$

$$\frac{\partial J_2}{\partial \sigma_{11}} = \frac{1}{3} (2\sigma_{11} - \sigma_{22} - \sigma_{33}) \tag{A.13}$$

$$\frac{\partial J_2}{\partial \sigma_{22}} = \frac{1}{3} (2\sigma_{22} - \sigma_{33} - \sigma_{11}) \tag{A.14}$$

$$\frac{\partial J_2}{\partial \sigma_{33}} = \frac{1}{3} (2\sigma_{33} - \sigma_{11} - \sigma_{22}) \tag{A.15}$$

$$\frac{\partial J_2}{\partial \sigma_{12}} = 2\sigma_{12} \tag{A.16}$$

$$\frac{\partial J_3}{\partial \sigma_{11}} = \frac{1}{3} (\sigma_{11}\sigma_{11} + 2\sigma_{22}\sigma_{33} + \sigma_{12}\sigma_{12}) \tag{A.17}$$

$$\frac{\partial J_3}{\partial \sigma_{22}} = \frac{1}{3} (\sigma_{22}\sigma_{22} + 2\sigma_{11}\sigma_{33} + \sigma_{12}\sigma_{12}) \tag{A.18}$$

$$\frac{\partial J_3}{\partial \sigma_{33}} = \frac{1}{3} (2\sigma_{22}\sigma_{11} - 2\sigma_{12}\sigma_{12}) \tag{A.19}$$

$$\frac{\partial J_3}{\partial \sigma_{12}} = -2\sigma_{33}\sigma_{12} \tag{A.20}$$

APPENDIX B
CAP MODEL CONSTITUTIVE MATRIX

This appendix derives implicitly the elastic-plastic constitutive matrix for the Cap model. The matrix is obtained from equation 4.26, 4.38, 4.39, 4.40 and 4.41 by proper mathematical manipulation.

$$C^{ep}{}_{ijrs} = C_{ijrs} - \frac{C_{ijkl} \frac{\partial F}{\partial \sigma_{kl}} \frac{\partial F}{\partial \sigma_{mn}} C_{mnr}}{\frac{\partial F}{\partial \sigma_{mn}} C_{mn}{}_{uv} \frac{\partial F}{\partial \sigma_{uv}} - \frac{\partial F}{\partial k} \frac{\partial k}{\partial \epsilon^p{}_{mn}} \frac{\partial F}{\partial \sigma_{mn}}} \quad (4.26)$$

$$F_f(l_1, \sqrt{J_2}) = A - \theta l_1 - C e^{Bl_1} \cdot \sqrt{J_2} = 0 \quad (4.38)$$

$$F_c(l_1, \sqrt{J_2}, k) = \frac{1}{R} \{ [X(k) - L(k)]^2 - [J_1 - L(k)]^2 \}^{1/2} \cdot \sqrt{J_2} = 0 \quad (4.39)$$

$$X(k) = k - R F_f(k) \quad (4.40a)$$

$$L(k) = k \quad \text{if } k < 0 \quad (4.40b)$$

$$L(k) = 0 \quad \text{if } k \geq 0 \quad (4.40c)$$

$$k = \epsilon^p{}_{kk} = W \{ \exp[D X(k) - 1] \} \quad (4.41)$$

For the plane-strain case, equation 4.26 can be expressed in matrix form as

$$[C^{ep}]_{4x4} = [C]_{4x4} - \frac{[C]_{4x4} \{DF\}_{4x1} \{DF\}_{4x1}^T [C]_{4x1}}{\{DF\}_{1x4}^T [C]_{4x4} \{DF\}_{4x1} + \frac{\partial F}{\partial k} \{DK\}_{1x4}^T \{DF\}_{4x1}} \quad (B.1)$$

where

$$[C] = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & 0 & \nu \\ \nu & 1-\nu & 0 & \nu \\ 0 & 0 & \frac{1-2\nu}{2} & 0 \\ \nu & \nu & 0 & 1-\nu \end{bmatrix} \quad (B.2)$$

$$\{DF\}^T_{1 \times 4} = \left\{ \frac{\partial F}{\partial \sigma_{11}}, \frac{\partial F}{\partial \sigma_{22}}, \frac{\partial F}{\partial \sigma_{12}}, \frac{\partial F}{\partial \sigma_{33}} \right\} \quad (B.3)$$

$$\{DK\}^T_{1 \times 4} = \left\{ \frac{\partial k}{\partial \varepsilon P_{11}}, \frac{\partial k}{\partial \varepsilon P_{22}}, \frac{\partial k}{\partial \varepsilon P_{12}}, \frac{\partial k}{\partial \varepsilon P_{33}} \right\} \quad (B.4)$$

Depending on the stress state, the partials in equation B.3 are found by differentiating equation 4.38 or 4.39. For the case where the stress state is on the failure surface,

$$\frac{\partial F}{\partial \sigma_{11}} = \frac{\partial F}{\partial I_1} \frac{\partial I_1}{\partial \sigma_{11}} + \frac{\partial F}{\partial J_2} \frac{\partial J_2}{\partial \sigma_{11}} \quad (B.5)$$

$$\frac{\partial F}{\partial \sigma_{22}} = \frac{\partial F}{\partial I_1} \frac{\partial I_1}{\partial \sigma_{22}} + \frac{\partial F}{\partial J_2} \frac{\partial J_2}{\partial \sigma_{22}} \quad (B.6)$$

$$\frac{\partial F}{\partial \sigma_{12}} = \frac{\partial F}{\partial I_1} \frac{\partial I_1}{\partial \sigma_{12}} + \frac{\partial F}{\partial J_2} \frac{\partial J_2}{\partial \sigma_{12}} \quad (B.7)$$

$$\frac{\partial F}{\partial \sigma_{33}} = \frac{\partial F}{\partial I_1} \frac{\partial I_1}{\partial \sigma_{33}} + \frac{\partial F}{\partial J_2} \frac{\partial J_2}{\partial \sigma_{33}} \quad (B.8)$$

where

$$\frac{\partial F}{\partial I_1} = -\theta - BCe^{BI_1} \quad (B.9)$$

$$\frac{\partial F}{\partial J_2} = -\frac{1}{2\sqrt{J_2}} \quad (B.10)$$

$$\frac{\partial I_1}{\partial \sigma_{11}} = \frac{\partial I_1}{\partial \sigma_{22}} = \frac{\partial I_1}{\partial \sigma_{33}} = 1 \quad (B.11)$$

$$\frac{\partial I_1}{\partial \sigma_{12}} = 0 \quad (B.12)$$

$$\frac{\partial J_2}{\partial \sigma_{11}} = \frac{1}{3} (2\sigma_{11} - \sigma_{22} - \sigma_{33}) \quad (B.13)$$

$$\frac{\partial J_2}{\partial \sigma_{22}} = \frac{1}{3} (2\sigma_{22} - \sigma_{33} - \sigma_{11}) \quad (B.14)$$

$$\frac{\partial J_2}{\partial \sigma_{33}} = \frac{1}{3} (2\sigma_{33} - \sigma_{11} - \sigma_{22}) \quad (B.15)$$

$$\frac{\partial J_2}{\partial \sigma_{12}} = 2\sigma_{12} \quad (B.16)$$

For the case where the stress state is on the hardening cap surface,

$$\frac{\partial F}{\partial \sigma_{11}} = \frac{\partial F}{\partial I_1} \frac{\partial I_1}{\partial \sigma_{11}} + \frac{\partial F}{\partial J_2} \frac{\partial J_2}{\partial \sigma_{11}} + \frac{\partial F}{\partial k} \frac{\partial k}{\partial \sigma_{11}} \quad (B.17)$$

$$\frac{\partial F}{\partial \sigma_{22}} = \frac{\partial F}{\partial I_1} \frac{\partial I_1}{\partial \sigma_{22}} + \frac{\partial F}{\partial J_2} \frac{\partial J_2}{\partial \sigma_{22}} + \frac{\partial F}{\partial k} \frac{\partial k}{\partial \sigma_{22}} \quad (B.18)$$

$$\frac{\partial F}{\partial \sigma_{12}} = \frac{\partial F}{\partial I_1} \frac{\partial I_1}{\partial \sigma_{12}} + \frac{\partial F}{\partial J_2} \frac{\partial J_2}{\partial \sigma_{12}} + \frac{\partial F}{\partial k} \frac{\partial k}{\partial \sigma_{12}} \quad (B.19)$$

$$\frac{\partial F}{\partial \sigma_{33}} = \frac{\partial F}{\partial I_1} \frac{\partial I_1}{\partial \sigma_{33}} + \frac{\partial F}{\partial J_2} \frac{\partial J_2}{\partial \sigma_{33}} + \frac{\partial F}{\partial k} \frac{\partial k}{\partial \sigma_{33}} \quad (B.20)$$

$$\frac{\partial F}{\partial I_1} = \frac{-[I_1 - L(k)]}{R \{ [X(k) - L(k)]^2 + [I_1 - L(k)]^2 \}^{1/2}} \quad (B.21)$$

$$\frac{\partial k}{\partial \sigma_{11}} = \frac{\partial k}{\partial \sigma_{22}} = \frac{\partial k}{\partial \sigma_{33}} = \frac{\partial k}{\partial \sigma_{12}} = 0 \quad (B.22)$$

$\partial F/\partial k$ will be derived in next paragraph. The rest of partials can be found from equation B.10 to B.16.

$\partial F/\partial k$ can be found by differentiating equation 4.39 and 4.40.

$$\begin{aligned} \frac{\partial F}{\partial k} &= \frac{\partial F}{\partial X(k)} \frac{\partial X(k)}{\partial k} + \frac{\partial F}{\partial L(k)} \frac{\partial L(k)}{\partial k} \\ &= \frac{[X(k) - L(k)]}{R \{ [X(k) - L(k)]^2 - [I_1 - L(k)]^2 \}^{1/2}} [1 + R\theta + RCBe^{BL(k)}] + \\ &\quad \frac{[I_1 - L(k)]}{R \{ [X(k) - L(k)]^2 - [I_1 - L(k)]^2 \}^{1/2}} \end{aligned} \quad (B.23)$$

The Partial in equation B.4 can be found by differentiating equation 4.41 and 4.40.

$$\begin{aligned} \frac{\partial k}{\partial \varepsilon_{11}} &= \frac{\partial k}{\partial \varepsilon_{kk}} \frac{\partial \varepsilon_{kk}}{\partial \varepsilon_{11}} \\ &= \frac{\partial k}{X(k)} \frac{\partial X(k)}{\partial \varepsilon_{kk}} \times 1 \\ &= \frac{1}{[1 + R\theta + RCBe^{BL(k)}]} \times \frac{1}{WDe^{DX(k)}} \end{aligned} \quad (B.24)$$

$$\frac{\partial k}{\partial \varepsilon_{22}} = \frac{\partial k}{\partial \varepsilon_{33}} = \frac{\partial k}{\partial \varepsilon_{11}} \quad (B.25)$$

$$\frac{\partial k}{\partial \varepsilon_{12}} = 0 \quad (B.26)$$

It should be noted that the second term in the denominator of equation B.1 is only needed when the stress state is on the Cap surface.

APPENDIX C INPUT GUIDE

Conventions

To each functional block corresponds a number of records to be read:

- (1) a header record containing the name of a functional block;
- (2) a record with parameters, if needed;
- (3) additional records containing other quantities, if necessary.

All the header records use the same format:

Variable	Columns	Format	Default	Description
BLOC M	1-4 5-10	A4 I6	- 0	Name of block to be executed Parameter controlling amount of output printing (M=0, 1, 2, or 3)

In general, all integer variables are read under I5 format, and real variables, under F10.0 format.

Block 'IMAG'

Function: Output of all inputted quantities (optional, but must be the first block if used).

Input: one record with the word 'IMAG'.

Block 'COMT'

Function: Output of comments (optional, but can be used at any time).

Input: (1) one record with the word 'COMT'
(2) comment records (maximum length of 80 characters), must be terminated with a blank record.

Block 'COOR'

Function: Reading of coordinates and number of degrees of freedom of all the nodes.

Input: (1) one record with the word 'COOR'

(2) one record of parameters;

Variable	Columns	Format	Default	Description
NNT	1-5	I5	20	Maximum number of nodes
NDLN	6-10	I5	2	Maximum number of DOF per node
NDIM	11-15	I5	2	Number of dimensions (1, 2, or 3)
FAC(1)	16-25	F10.0	1.0	Scaling factor in direction x
FAC(2)	26-35	F10.0	1.0	Scaling factor in direction y
FAC(3)	36-45	F10.0	1.0	Scaling factor in direction z
YWAT	46-55	F10.0	0.0	Y coordinate of the existing water table
(3) series of records for nodes; terminated by a record on which IN1 .LE. 0, each record may generate many nodes.				
IN1	1-5	I5	-	Number of the first node to be generated
X1(1)	6-15	F10.0	-	x coordinate
X1(2)	16-25	F10.0	-	y coordinate
X1(3)	26-35	F10.0	-	z coordinate
IN2	36-40	I5	IN1	Number of the last node to be generated
X2(1)	41-50	F10.0	X1(1)	x coordinate
X2(2)	51-60	F10.0	X1(2)	y coordinate
X2(3)	61-70	F10.0	X1(3)	z coordinate
INCR	71-75	I5	1	Pitch of node numbers
IDLN	76-80	I5	NDLN	Number of DOF generated if different from default (NDLN)

Remarks

- (A) The number of degrees of freedom of a node must be coherent with the number of freedoms at a node for a given element
- (B) If the nodes are all entered with no generation parameters, IN2 to IDLN (columns 36-80) are left null.

Block 'COND'

Function: Reading of boundary conditions (required).

- Input:
- (1) one record with the word 'COND';
 - (2) groups of two records terminated by a blank record:
 - a. one header record for each group of boundary conditions;

Variable	Columns	Format	Default	Description
ICOD	1-10	10I1	-	For each degree of freedom (maximum 10) 0 if free 1 if prescribed
V	11-80	7F10.0	-	Values for prescribed DOF in the same order as ICOD, the number of values read in V equal to maximum degrees of freedom per node(NDLN)
KV	1-80	16I5	-	Node numbers which must be terminated by zero. If all 16 values are non-zero, reading will continue on the following record.

Remark

The list of values V can be continued with additional records of format (10X,
7E10.0). The list of nodal numbers KV can also be continued with additional records of
format (16I5).

Block 'PREL'

Function: Reading of element material properties (required).

- Input:
- (1) one record with the word 'PREL';
 - (2) one record of parameters;

Variable	Columns	Format	Default	Description
NGPE	1-5	I5	0	Number of groups of element material properties
NPRE	6-10	I5	0	Number of linear properties per group
NPRM	11-15	I5	0	Number of nonlinear properties per group
ROWA	16-25	F10.0	62.4	Unit weight of water
BKWA	26-35	F10.0	2.50D5	Bulk modulus of water
ATMP	36-45	F10.0	14.7	Atmospheric pressure

- (3) series of records of group properties terminated by a record in which
IGPE .LE. 0.

IGPE	1-5	I5	-	Group number
MODEL	6-8	I3	0	Nonlinear model number 0 = Linear 1 = Various modulus 2 = Mohr-Columb plasticity 3 = Drucker-Prager plasticity 4 = Sandler-Cap plasticity
IFEL	9-10	I2	0	Interface element number 0 = Regular element 1 = Interface element
V1	11-80	7F10.0	-	Successive values of properties

Remarks

- (A) NPRE must be equal to the maximum number of linear material properties required by the element used.
- (B) NPRM must be equal to the maximum number of parameters of the models required by the element used in the problem.
- (C) If NPRE+NPRM .GT. 7, the list of properties V1 is continued with additional records of format (5X,7F10.0).

Block 'ELEM'

Function: Reading of elements (connectivities) (required).

Input: (1) one record with the word 'ELEM';

(2) one record of parameters;

Variable	Columns	Format	Default	Description
NELT	1-5	I5	-	Maximum number of elements
NNEL	6-10	I5	8	Maximum number of nodes per element
NKEL	11-15	I5	1	Number of problem type by default
NTPE	16-20	I5	1	Number of element type by default
NSDG	21-25	I5	0	= 0 if no saturated elements = 1 if saturated elements
NLAG	26-30	I5	0	= 0 for small deformation and small strain = 1 for large deformation and large strain
NSYM	31-35	I5	0	= 0 if matrix KG symmetrical = 1 if matrix KG non-symmetrical
NIDENT	36-40	I5	0	= 1 if all element stiffness matrices are identical

(3) element records terminated by a record in which IEL.LE.0. Each record may generate more than one element.

IEL	1-5	I5	-	Number of the first element
IGEN	6-10	I5	1	Total number of elements to be generated
INCR	11-15	I5	1	Nodal number pitch for automatic generation
IKEL	16-20	I5	NKEL	= 1 for plane stress = 2 for plane strain = 3 for axisymmetrical
ITPE	21-25	I5	NTPE	Element type number if different from NTPE
IGPE	26-30	I5	1	Element properties group number
ISDE	31-35	I5	NSDE	= 0 if not saturated = 1 if saturated
KNE	36-80	9I5	-	Nodal numbers of the element, terminated by a zero node number (order must follow convention)

Block 'CONC'

Function: Reading of concentrated loads (optional).

- Input:
- (1) one record with the word 'CONC';
 - (2) groups of two records terminated by a blank record:
 - a. one header record per group of forces;

Variable	Columns	Format	Default	Description
IG V	1-5 6-75	I5 7F10.0	-	Group number Load values for each degree of freedom
KV	1-80	16I5	-	Node numbers so loaded, terminated with a zero node number

Remarks

- (A) List V can be continued on additional records of format (5X,7F10.0).
- (B) List KV can be continued on additional records of format (16I5).

Block 'BODY'

Function: Computation and assemblage of distributed loads (optional).

- Input: one record with the word 'BODY'.

Block 'INSI'

Function: Computation and assemblage of distributed loads in a problem with construction sequences (required).

- Input: one record with the word 'INSI'.

Block 'LOAD'

Function: Reading of concentrated loads in a problem with construction sequences (optional).

Input: (1) one record with the word 'LOAD';
 (2) see input (2) in BLOCK 'CONC'.

Block 'DWAT'

Function: Computation and assemblage of equivalent nodal loads due to dewatering in a problem with construction sequences (optional).

Input: (1) one record with the word 'DWAT';
 (2) one record of parameters

Variable	Columns	Format	Default	Description
NELS	1-5	I5	-	Number of elements dewatered (3) series of records for elements; terminated by a record on which IEL .LE. 0.
IEL	1-5	I5	-	Number of the first element
IGEN	6-10	I5	-	Number of elements to be generated
INCR	11-15	I5	-	Pitch of the element numbers

Block 'EXCA'

Function: Computation and assemblage of equivalent nodal loads due to excavation in a problem with construction sequences (optional).

Input: (1) one record with the word 'EXCA';
 (2) one record of parameters;

Variable	Columns	Format	Default	Description
NELS	1-5	I5	-	Number of elements excavated (3) one record with the "air" element group number;
IGPE	6-10	I5	-	Group number of "air" element (4) series of records for elements; terminated by a record on which IEL .LE. 0;
IEL	1-5	I5	-	Number of the first element
IGEN	6-10	I5	-	Number of elements to be generated
INCR	11-15	I5	-	Pitch of the element numbers

Remark

The elastic modulus for the "air" element is recommended to have a value of 10E-5 when a problem deals with excavation and embankment.

Block 'EMBA'

Function: Computation and assemblage of equivalent nodal loads due to embankment in a problem with construction sequences (optional).

Input: (1) one record with the word 'EMBA';
 (2) one record of parameters;

Variable	Columns	Format	Default	Description
NELS	1-5	I5	-	Number of elements added
	(3)			series of records for elements; terminated by a record on which
				IEL .LE. 0;
IEL	1-5	I5	-	Number of the first element
IGEN	6-10	I5	-	Number of elements to be generated
INCR	11-15	I5	-	Pitch of the element numbers
IGPE	16-20	I5	-	Group number of element property

Block 'FWAT'

Function: Computation and assemblage of equivalent nodal loads due to the raise of water table in a problem with construction sequences (optional).

- Input:
- (1) one record with the word 'FWAT'
 - (2) one record of parameters;

Variable	Columns	Format	Default	Description
NELS	1-5	I5	-	Number of elements saturated
	(3)			series of records for elements; terminated by a record on which
				IEL .LE. 0.
IEL	1-5	I5	-	Number of the first element
IGEN	6-10	I5	-	Number of elements to be generated
INCR	11-15	I5	-	Pitch of the element numbers

Remark

If the water table rises high enough, the water pressure may act against the structure built in the soil medium. In this case, it is necessary to find the equivalent nodal load acting on the structure due to the water pressure and the loading vector can be generated using Block 'FWAT' and Block 'CONC'.

Block 'BARA'

Function: Installation of bar elements in a problem with construction sequences (optional).

- Input:
- (1) one record with the word 'BARA';
 - (2) one record of parameters;

Variable	Columns	Format	Default	Description
NELS	1-5	I5	-	Number of elements added
	(3)			series of records for elements; terminated by a record on which IEL .LE. 0.
IEL	1-5	I5	-	Element number
IGPE	6-10	I5	-	Group number of element property

Remark

This is the only loading block that does not come with an execution block.

Block 'BARD'

Function: Removal of bar elements in a problem with construction sequences(optional).

- Input:
- (1) one record with the word 'BARD';
 - (2) one record of parameters;

Variable	Columns	Format	Default	Description
NELS	1-5	I5	-	Number of elements deleted
	(3)			series of records for elements; terminated by a record on which IEL .LE. 0.
IEL	1-5	I5	-	Element number

Block 'LINM'

Function: Assemblage and solution of an in-core linear problem (optional).

Input: one record with the word 'LINM'.

Block 'NLIN'

Function: Assemblage and solution of an in-core nonlinear problem (optional).

Input: (1) one record with the word 'NLIN';
 (2) one record of parameters

Variable	Columns	Format	Default	Description
DPAS	1-10	F10.0	0.2	Load increment
NPAS	11-15	I5	1	Number of identical load increments
NITER	15-20	I5	5	Number of iterations per step
IMETH	21-25	I5	4	Iteration method: 1 - computation of KG for each iteration 2 - KG constant 3 - computation of KG at start of each step
EPSDL	31-40	F10.0	0.01	Admissible error of norm
OMEGA	41-50	F10.0	1.0	Over-relaxation factor
IELS	51-55	I5	0	Index for printing specified element solutions 0 = no 1 = yes
IEQL	56-60	I5	0	Index for computing equivalent nodal loads of the specified elements 0 = no 1 = yes
				(3) one record of element numbers if IELS = 1.
NSOL	1-80	16I5	-	Specified element numbers

Block 'LIND'

Function: Assemblage and solution of an out of core linear problem with a partitioned matrix stored in a mass storage device (optional).

Input: one record with the word 'LIND'.

Block 'NLND'

Function: Assemblage and solution of an out of core nonlinear problem with a partitioned matrix stored in a mass storage device (optional).

Input: (1) one record with the word 'NLND';
 (2) see input (2) in BLOCK 'NLIN'.

Block 'PRNT'

Function: Printing solutions for specified elements (optional).

Input: (1) one record with the word 'PRNT';
 (2) one record of parameters;

Variable	Columns	Format	Default	Description
NELS	1-5	I5	-	Number of elements to be printed
	(3)			series of records for elements; terminated by a record on which
				IEL .LE. 0.
IEL	1-5	I5	-	Number of the first element
IGEN	5-10	I5	-	Number of elements to generate
INCR	11-15	I5	-	Pitch of the element numbers

Block 'PLOT'

Function: Creating a output data file to draw the deformed structure (optional).

Input: one record with the word 'PLOT'.

Block 'STOP'

Function: End (required).

Input: one record with the word 'STOP'.

APPENDIX D FORTRAN LISTING OF COMPUTER PROGRAM FEMCON

```

C=====
C   FEMCON88 (2-D FINITE ELEMENT PROGRAM)      HSI-CHI YANG, July 1988
C
C   FEMCON (FINITE ELEMENT METHOD WITH CONSTRUCTION SEQUENCES) IS
C   INTENDED FOR 2-D NONLINEAR ANALYSES OF SOIL-STRUCTURE INTERACTION
C   PROBLEMS WITH OR WITHOUT CONSTRUCTION SEQUENCES. IT CONSIDERS
C   BOTH MATERIAL AND GEOMETRIC NONLINEARITIES.
C
C   . MAIN PROGRAM
C
C=====
IMPLICIT REAL*8(A-H,O-Z)
CHARACTER*4 BLOC,BLOCS
COMMON/ALLO/NVA,IVA,IVAMAX,NREEL,NTBL
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
COMMON VA(20000)
DIMENSION BLOCS(28)
DATA BLOCS/'IMAG','COMT','COOR','COND','PREL','ELEM','CONC',
1      'BODY','INSI','DWAT','EXCA','EMBA','BARA','BARD',
2      'LOAD','FWAT','LINM','NLIN','LIND','NLND','PRNT',
3      'PLOT','....','....','....','....','....','STOP'/
DATA NB/28/
C
OPEN(MR,FILE='INPUT.DAT')
OPEN(MP,FILE='OUTPUT.DAT',STATUS='NEW')
OPEN(M1,FILE='M1.DAT',STATUS='NEW')
REWIND MR
C----- LENGTH OF BLANK COMMON IN REAL WORDS (TABLE VA)
NVA=20000
C----- HEADING
WRITE(MP,2000)
2000 FORMAT('1',30X,'UFRAME88'/28X,'HSI-CHI YANG'/23X,22('-')//)
C----- READ BLOCK TITLE
10  READ(MR,1000) BLOC,M
1000 FORMAT(A4,I6)
C----- SEARCH FOR THE BLOCK TO BE EXECUTED
DO 20 I=1,NB
IF(BLOC.EQ.BLOCS(I)) GO TO 30
20 CONTINUE
WRITE(MP,2010)
2010 FORMAT(' ** ERROR, MISSING BLOCK CALLING CARD')
GO TO 10
30  GO TO (110,120,130,140,150,160,170,
1      180,190,200,210,220,230,240,
2      250,260,270,280,290,300,310,
3      320,330,340,350,360,370,999),I
C----- BLOCK TO PRINT IMAGES OF DATA CARDS           'IMAG'
110 CALL BLIMAG
GO TO 10
C----- BLOCK TO READ AND PRINT COMMENTS            'COMT'
120 CALL BLCOMT
GO TO 10
C----- BLOCK TO READ NODAL POINTS COORDINATES     'COOR'

```

```

130 CALL BLCOOR
      GO TO 10
C----- BLOCK TO READ BOUNDARY CONDITIONS      'COND'
140 CALL BLCOND
      GO TO 10
C----- BLOCK TO READ ELEMENT PROPERTIES      'PREL'
150 CALL BLPREL
      GO TO 10
C----- BLOCK TO READ ELEMENT DATA            'ELEM'
160 CALL BLELEM
      GO TO 10
C----- BLOCK TO READ CONCENTRATED LOADS(NON-CONSTRUCTION) 'CONC'
170 CALL BLCONC
      GO TO 10
C----- BLOCK TO READ BODY LOADS(NON-CONSTRUCTION)      'BODY'
180 CALL BLBODY
      GO TO 10
C----- BLOCK TO READ INSITU DATA            'INSI'
190 CALL BLINSI
      GO TO 10
C----- BLOCK TO READ DEWATER DATA          'DWAT'
200 CALL BLDWAT
      GO TO 10
C----- BLOCK TO READ EXCAVATION DATA        'EXCA'
210 CALL BLEXCA
      GO TO 10
C----- BLOCK TO READ EMBANKMENT DATA        'EMBA'
220 CALL BLEMBA
      GO TO 10
C----- BLOCK TO READ ADDING-BARS DATA       'BARA'
230 CALL BLBARA
      GO TO 10
C----- BLOCK TO READ DELETING-BARS DATA     'BARD'
240 CALL BLBARD
      GO TO 10
C----- BLOCK TO READ CONCENTRATED LOADING DATA 'LOAD'
250 CALL BLLOAD
      GO TO 10
C----- BLOCK TO READ RAISING-WATER-TABLE DATA 'FWAT'
260 CALL BLFWAT
      GO TO 10
C----- BLOCK FOR IN CORE ASSEMBLING AND LINEAR SOLUTION 'LINM'
270 CALL BLLINM
      GO TO 10
C----- BLOCK FOR IN CORE ASSEMBLING AND NONLINEAR SOLUTION 'NLIN'
280 CALL BLNLIN
      GO TO 10
C----- BLOCK FOR ON DISK ASSEMBLING AND LINEAR SOLUTION 'LIND'
290 CALL BLLIND
      GO TO 10
C----- BLOCK FOR ON DISK ASSEMBLING AND NONLINEAR SOLUTION 'NLND'
300 CALL BLNLND
      GO TO 10
C----- BLOCK TO PRINT SOLUTIONS FOR SPECIFIED ELEMENTS 'PRNT'

```

```

310 CALL BLPRNT
      GO TO 10
C----- BLOCK TO PLOT GEOMETRIC (DEFORMED) MESH          'PLOT'
320 CALL BLPLOT
      GO TO 10
330 CONTINUE
      GO TO 10
340 CONTINUE
      GO TO 10
350 CONTINUE
      GO TO 10
360 CONTINUE
      GO TO 10
370 CONTINUE
      GO TO 10
C----- END OF PROBLEM                                     'STOP'
999 WRITE(MP,2020) IVAMAX,NVA
2020 FORMAT(//' END OF PROBLEM, ',I10,' UTILIZED REAL WORDS OVER ',I10)
      STOP
      END
      BLOCK DATA
C=====
C   INITIALIZE LABELLED COMMONS
C=====
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/COOR/NDIM,NNT,NDLN,NDLT
      COMMON/COND/NCLT,NCLZ,NCLNZ
      COMMON/PREL/NGPE,NPRE,NPRM,NMNL,NHIS
      COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
      1 NPEM,NELS
      COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDEL
      COMMON/RGDT/IEL,IKEL,ITPE,ITPE1,ISDE,IMODEL,IFEL,IDELE,ICE,INEL,
      1 IPG,ICODE,IMATD,INSE,INPE,IDELO,INELO,IPGO,IRES
      COMMON/NLIN/EPSDL,OMEGA,DPAS,NPAS,NITER,ITER,IMETH,ILAG,INLN
      COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
      COMMON/ALLO/NVA,IVA,IVAMAX,NREEL,NTBL
      COMMON/LOCA/LCORG,LDLNC,LNEQ,LDIMP,LCODE,LINTF,LPREG,LLD,LLOCE,
      1 LCORE,LNE,LPREE,LNPEG,LSDEG,LDLE,LKE,LFE,LKGS,LKGD,LKGI,LFG,
      2 LRES,LDLG,LNELS,LEB,LPB
      COMMON/SEQU/NSEQ,NSEQ0,ISEQ,IASSEL,IBAR
      COMMON/TVL2/FAC(3)
      COMMON/TVL3/YWAT,ROWA,BKWA,ATMP
      DIMENSION LXX(26)
      EQUIVALENCE (LXX(1),LCORG)
C----- COMMON /COOR/
      DATA NNT/20/,NDLN/2/,NDIM/2/
C----- COMMON /PREL/
      DATA NGPE/0/,NPRE/0/,NPRM/0/,NMNL/0/,NHIS/0/
C----- COMMON /ELEM/
      DATA NNEL/8/,NKEL/1/,NTPE/1/,NSDG/0/,NLAG/0/,NIDENT/0/,NELS/0/
C----- COMMON /ASSE/
      DATA NSYM/0/
C----- COMMON /RGDT/
      DATA ITPE1/0/,ISDE/0/

```

```

C----- COMMON /NLIN/
      DATA EPSDL/1.D-2/,OMEGA/1.D0/,DPAS/.2D0/,NPAS/1/,NITER/5/,IMETH/4/
C----- COMMON /NUMB/
      DATA M1/1/,ME/2/,M3/3/,M4/4/,MR/11/,MP/12/,M7/7/,MS/8/,M9/9/,
     1 MH/10/
C----- COMMON /ALLO/
      DATA IVA/1/,IVAMAX/1/,NTBL/28/
C----- DEFINE HERE THE NUMBER OF INTEGERS CONTAINED IN A REAL
C         FOR THE COMPUTER EMPLOYED
C         EXAMPLES: AT&T SIMPLE PRECISION   NREEL.EQ.1
C                  AT&T DOUBLE PRECISION   NREEL.EQ.2
C                  CDC                   NREEL.EQ.1
      DATA NREEL/2/
C----- COMMON /LOCA/
      DATA LXX/26*1/
C----- COMMON /SEQU/
      DATA NSEQ/0/,NSEQ0/0/,ISEQ/0/,IASSEL/1/,IBAR/0/
C----- COMMON /TVL2/ AND /TVL3/
      DATA FAC/3*1.D0/,YWAT/0.D0/,ROWA/62.4D0/,BKWA/2.5D5/,ATMP/14.7D0/
      END
      SUBROUTINE ERROR(IERR,I1,I2,INIV)
C=====
C   PRINT ERROR MESSAGES FOR BLOCKS READING DATA
C=====
      COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
C-----
C----- BLOCK 'COOR'
      IF(IERR.GT.19) GO TO 200
      IE=IERR-10
      GO TO (110,120,130,140,150,160,160,180),IE
110  WRITE(MP,2110)I1,I2
2110 FORMAT(' *** ERROR, FIRST NODE NUMBER(',I4,') IS GREATER THAN NNT=
     1',I4)
      GO TO 900
120  WRITE(MP,2120)I1,I2
2120 FORMAT(' ** ERROR, SECOND NODE NUMBER(',I4,') IS GREATER THAN NNT=
     1',I4)
      GO TO 900
130  WRITE(MP,2130)I1,I2
2130 FORMAT(' ** ERROR, NODAL NUMBER OF D.O.F.(',I4,') IS GREATER THAN
     1NDLN=',I4)
      GO TO 900
140  WRITE(MP,2140)
2140 FORMAT(' ** ERROR, FIRST AND SECOND NODE NUMBERS ARE INCOMPATIBLE
     1WITH THE GENERATION PARAMETER')
      GO TO 900
150  WRITE(MP,2150)I1
2150 FORMAT(' ** ERROR, NODE ',I4,' IS DEFINED MORE THAN ONCE')
      GO TO 900
160  WRITE(MP,2160)I1
2160 FORMAT(' ** ERROR, NODE ',I4,' IS NOT DEFINED')
      GO TO 900
180  WRITE(MP,2180)I2,I1
2180 FORMAT(' ** ERROR, GENERATED NODES NUMBER(',I4,') IS LESS THAN NNT

```

```

1=',I4)
GO TO 900
200 IF(IERR.GT.29) GO TO 300
GO TO 900
C----- BLOCK 'COND'
300 IF(IEER.GT.39) GO TO 400
IE=IERR-30
GO TO (900,320,900),IE
320 WRITE(MP,320)I1,I2
2320 FORMAT(' ** ERROR, NODE NUMBER(',I4,) IS GREATER THAN
1NNNT=',I4)
GO TO 900
C----- BLOCK 'PREL'
400 IF(IERR.GT.49) GO TO 500
IE=IERR-40
GO TO (410,900),IE
410 WRITE(MP,2410)I1,I2
2410 FORMAT(' ** ERROR, GROUP NUMBER (',I3,) IS GREATER THAN NGPE=',I3
1)
GO TO 900
C----- BLOCK 'ELEM'
500 IF(IERR.GT.59) GO TO 900
IE=IERR-50
GO TO (510,900,530,540,550,560,570),IE
510 WRITE(MP,2510)I1,I2
2510 FORMAT(' ** ERROR, NUMBER OF NODES (',I3,) IS GREATER THAN NNEL='
1,I3)
GO TO 900
530 WRITE(MP,2530)I1,I2
2530 FORMAT(' ** ERROR, PROPERTY NUMBER (',I3,) IS GREATER THAN NGPE='
1,I3)
GO TO 900
540 WRITE(MP,2540)I1,I2
2540 FORMAT(' ** ERROR, EL. BEHAVIOR # (',I3,) IS GREATER THAN NSDG='
1,I3)
GO TO 900
550 WRITE(MP,2550)I1,I2
2550 FORMAT(' ** ERROR, ELEMENT NUMBER (',I4,) IS GREATER THAN NELT='
1I4)
GO TO 900
560 GO TO 320
570 WRITE(MP,2570)I1,I2
2570 FORMAT(' ** ERROR, NUMBER OF ELEMENTS (',I4,) IS GREATER THAN NEL
1T=',I4)
C----- END
900 I1=I2
IF(INIV.GE.2) STOP
RETURN
END
SUBROUTINE ESPACE(ILONG,IREEL,TBL,IDEBUG)
=====
C      TO ALLOCATE A REAL OR INTEGER TABLE IN ARRAY VA
C      INPUT
C          ILONG           LENGTH OF THE TABLE TO BE ALLOCATED

```

```

C           (IN REAL OR INTEGER WORDS)
C      IREEL          TABLE TYPE :
C           .EQ.0      INTEGER
C           .EQ.1      REAL
C      TBL           NAME OF THE TABLE (A4)
C      OUTPUT
C      IDEB          TABLE TO BE ALLOCATED STARTS IN VA(IDEBUG)
C=====
C      IMPLICIT REAL*8(A-H,O-Z)
C      CHARACTER*4 TBL
C      COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
C      COMMON/ALLO/NVA,IVA,IVAMAX,NREEL,NTBL
C      COMMON VA(20000)
C      DIMENSION KA(40000)
C      EQUIVALENCE (VVA(1),KA(1))
C      DATA ZERO/0.D0/
C-----
C----- CALCULATE THE TABLE LENGTH IN REAL WORDS
C      ILGR=ILONG
C      IF(IREEL.EQ.0) ILGR=(ILONG+NREEL-1)/NREEL
C      IVA1=IVA+ILGR
C----- CHECK IF ENOUGH SPACE IS AVAILABLE
C      IF(IVA1.LE.NVA) GO TO 20
C..... AUTOMATIC EXTENSION OF THE BLANK COMMON IF CORRESPONDING
C      SYSTEM COMMAND EXIST ON THE COMPUTER USED
C      CALL EXTEND(IVA1,IERR)
C      IF(IERR.EQ.1) GO TO 10
C      NVA=IVA1
C      GO TO 20
C----- ALLOCATION ERROR (NOT ENOUGH SPACE)
10     WRITE(MP,2000) TBL,IVA1,NVA
2000   FORMAT(' **** ALLOCATION ERROR, TABLE ',A4/' REQUIRED SPACE:',I9,
1      REAL WORDS, AVAILABLE SPACE:',I9,' REAL WORDS')
      STOP
C----- ALLOCATE TABLE
20     IDEB=IVA+1
      IVA=IVA1
      IF(IVA.GT.IVAMAX) IVAMAX=IVA
      IF(M.GT.0) WRITE(MP,2010) TBL,IDEBUG,IVA1
2010   FORMAT(60X,'TABLE ',A4,' GOES FROM VA(',I7,') TO VA(',I7,')')
C----- INITIALIZE THE ALLOCATED TABLE TO ZERO
      I1=IDEBUG
      IF(IREEL.EQ.0) I1=(I1-1)*NREEL+1
      I2=I1+ILONG-1
      IF(IREEL.EQ.0) GO TO 40
      DO 30 I=I1,I2
30     VA(I)=ZERO
      RETURN
40     DO 50 I=I1,I2
50     KA(I)=0
      RETURN
      END
      SUBROUTINE DSPACE(IDEBUG,IREEL,TBL)
C=====

```

```

C      TO DELETE A TABLE FROM VA, FOLLOWED BY COMPACTING
C      INPUT
C          IDEB           FIRST POSITION OF TABLE TO BE FELETED
C          IREEL          TABLE TYPE (SEE ESPACE)
C          TBL            NAME OF THE TABLE (A4)
C=====
IMPLICIT REAL*8(A-H,O-Z)
CHARACTER*4 TBL
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
COMMON/ALLO/NVA,IVA,IVAMAX,NREEL,NTBL
COMMON/LOCA/LXX(26)
COMMON VA(20000)

C-----
C----- SEARCH FOR THE FIRST POSITION OF NEXT TABLE
I1=IVA+1
DO 10 I=1,NTBL
IF(LXX(I).LE.IDEB) GO TO 10
IF(LXX(I).LT.I1) I1=LXX(I)
10 CONTINUE
C----- SHIFT ALL TABLES AFTER THIS
ID=I1-IDEB
IF(I1.EQ.IVA+1) GO TO 40
DO 20 I=1,NTBL
IF(LXX(I).GT.IDEB) LXX(I)=LXX(I)-ID
20 CONTINUE
DO 30 I=I1,IVA
J=I-ID
30 VA(J)=VA(I)
C----- PRINT
40 IVA=IVA-ID
IF(M.GT.0) WRITE(MP,2000) TBL,ID,IDE
2000 FORMAT(60X,'DELETED TABLE ',A4,' COMPACTING ',I7,' REAL WORDS AFTE
1R VA(',I7,')')
RETURN
END
SUBROUTINE BLIMAG
C=====
C      TO CALL AND EXECUTE BLOCK 'IMAG'
C      TO PRINT OUT THE IMAGE OF DATA CARDS
C=====
IMPLICIT REAL*8(A-H,O-Z)
CHARACTER*4 CART
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
COMMON/TVL1/CART(20)
DATA ICARTM/40/
C-----
WRITE(*,3000)
3000 FORMAT(' BLIMAG')
WRITE(MP,2000)
2000 FORMAT(//,1X,'IMAGE OF DATA CARDS'/1X,28('='),/)
WRITE(MP,2005)
2005 FORMAT(
1 28X,'C O L U M N   N U M B E R',//,10X,'1','2',9X,'3',9X,'4',
2 9X,'5',9X,'6',9X,'7',9X,'8',//,1X,8('1234567890'),/,1X,80('-'))

```

```

ICART=0
ICART1=0
10 READ(MR,1000,END=30) CART
1000 FORMAT(20A4)
ICART=ICART+1
ICART1=ICART1+1
IF(ICART1.LE.ICARTM) GO TO 20
WRITE(MP,2010)
2010 FORMAT(1X,80(''),/,,1X,8('1234567890'),/,10X,'1',9X,'2',9X,'3',
1 9X,'4',9X,'5',9X,'6',9X,'7',9X,'8',/,28X,
2 'C O L U M N   N U M B E R')
WRITE(MP,2015)
2015 FORMAT('1',//)
WRITE(MP,2005)
ICART1=0
20 WRITE(MP,2020) CART
2020 FORMAT(1X,20A4)
GO TO 10
30 WRITE(MP,2010)
WRITE(MP,2030).
2030 FORMAT(///,29X,'E N D   O F   D A T A',/'1')
REWIND MR
READ(MR,1000) CART
RETURN
END
SUBROUTINE BLCOMT
C=====
C TO CALL AND EXECUTE BLOCK 'COMT'
C=====
IMPLICIT REAL*8(A-H,O-Z)
CHARACTER*4 BLANC,CART
COMMON/NUMB/M,M1,M2,M3,M4,MR,MP,M7,MS,M9,MH
COMMON/TVL1/CART(20)
DATA BLANC/'      '/
C-----
WRITE(*,3000)
3000 FORMAT(' BLCOMT')
WRITE(MP,2000)
2000 FORMAT(//' COMMENTS'/' ',8('')/')
C----- READ A COMMENT CARD
10 READ(MR,1000) CART
1000 FORMAT(20A4)
C----- SEARCH FOR A WHOLLY BLANK CARD
DO 20 I=1,20
IF(CART(I).NE.BLANC) GO TO 30
20 CONTINUE
RETURN
30 WRITE(MP,2010) CART
2010 FORMAT(1X,20A4)
GO TO 10
END
SUBROUTINE BLCOOR
C=====
C TO CALL BLOCK COOR

```

```

C      TO READ NODAL COORDINATES
C=====
IMPLICIT REAL*8(A-H,O-Z)
CHARACTER*4 TBL
COMMON/COOR/NDIM,NNT,NDLN,NDLT
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
COMMON/ALLO/NVA,IVA,IVAMAX,NREEL,NTBL
COMMON/LOCA/LCORG,LDLNC,LNEQ,LDIMP,LCODE,LINTF,LPREG,LLD,LLOCE,
1  LCORE,LNE,LPREE,LNPEG,LSDEG,LDLE,LKE,LFE,LKGS,LKGD,LKGI,LFG,
2  LRES,LDLG,LNELS,LEB,LPB
COMMON/TVL2/FAC(3)
COMMON/TVL3/YWAT,ROWA,BKWA,ATMP
COMMON VA(20000)
DIMENSION TBL(2),FAC1(3),IN(3)
DATA TBL/'CORG','DLNC',/ZERO/0.D0/
C-----
      WRITE(*,3000)
3000 FORMAT(' BLCOOR')
C----- BLOCK HEADING
      READ(MR,1000) IN,FAC1,Y
1000 FORMAT(3I5,4F10.0)
C----- DEFAULT OPTIONS
      IF(IN(1).GT.0) NNT=IN(1)
      IF(IN(2).GT.0) NDLN=IN(2)
      IF(IN(3).GT.0) NDIM=IN(3)
      DO 10 I=1,3
10    IF(FAC1(I).NE.ZERO) FAC(I)=FAC1(I)
      IF(Y.NE.ZERO) YWAT=Y
C----- PRINT BLOCK PARAMETERS
      WRITE(MP,2000) M,NNT,NDLN,NDIM,FAC(1),FAC(2),FAC(3),NVA
2000 FORMAT(//' INPUT OF NODES (M=',I2,')'//',21('*')/
1  15X,'MAX. NUMBER OF NODES          (NNT)=',I5/
2  15X,'MAX. NUMBER OF D.O.F. PER NODE (NDLN)=',I5/
3  15X,'DIMENSIONS OF THE PROBLEM     (NDIM)=',I5/
4  15X,'COORDINATE SCALE FACTORS     (FAC)=',E12.5/
4  15X,'                                ',E12.5/
4  15X,'                                ',E12.5/
5  15X,'WORKSPACE IN REAL WORDS      (NVA)=',I10)
C----- ALLOCATE SPACE
      IF(LCORG.EQ.1) CALL ESPACE(NNT*NDIM,1,TBL(1),LCORG)
      IF(LDLNC.EQ.1) CALL ESPACE(NNT+1,0,TBL(2),LDLNC)
C----- EXECUTE THE BLOCK
      CALL EXCOOR(VA(LCORG),VA(LDLNC))
      RETURN
      END
      SUBROUTINE EXCOOR(VCORG,KDLNC)
C=====
C      TO EXECUTE BLOCK 'COOR'
C      READ NODAL COORDINATES
C=====
IMPLICIT REAL*8(A-H,O-Z)
COMMON/COOR/NDIM,NNT,NDLN,NDLT
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
COMMON/TVL2/FAC(3)

```

```

DIMENSION X1(3),X2(3),VCORG(*),KDLNC(*)
DATA SPECL/1.23456789D31/
C----- INITIALIZE COORDINATES
      I1=(NNT-1)*NDIM+1
      DO 10 I=1,I1,NDIM
10    VCORG(I)=SPECL
C----- READ NODAL DATA CARDS
      IF(M.GT.0) WRITE(MP,2000)
2000 FORMAT('/* NODAL DATA CARDS'*)
20    READ(MR,1000) IN1,X1,IN2,X2,INCR,IDLN
1000 FORMAT(2(I5,3F10.0),2I5)
      IF(M.GT.0) WRITE(MP,2010) IN1,X1,IN2,X2,INCR,IDLN
2010 FORMAT(' >>>',2(I5,3E12.5),2I5)
      IF(IN1.LE.0) GO TO 60
C----- DECODE THE CARD
      IF(IN1.GT.NNT) CALL ERROR(11,IN1,NNT,0)
      IF(IN2.GT.NNT) CALL ERROR(12,IN2,NNT,0)
      IF(IN2.LE.0) IN2=IN1
      IF(IDLN.GT.NDLN) CALL ERROR(13,IDLN,NDLN,0)
      IF(IDLN.LE.0) IDLN=NDLN
      IF(INCR.LE.0) INCR=1
      I1=(IN2-IN1)/INCR
      I2=IN1+I1*INCR
      IF(I1.EQ.0) I1=1
      IF(IN2.NE.I2) CALL ERROR(14,IN2,IN2,0)
C----- GENERATE NODES BY INTERPOLATION
      DO 30 I=1,NDIM
      X1(I)=X1(I)*FAC(I)
      X2(I)=X2(I)*FAC(I)
30    X2(I)=(X2(I)-X1(I))/I1
      I1=0
      I2=(IN1-1)*NDIM+1
      I3=(INCR-1)*NDIM
      DO 50 IN=IN1,IN2,INCR
      KDLNC(IN+1)=IDLN
      IF(VCORG(I2).NE.SPECL) CALL ERROR(15,IN,IN,0)
      DO 40 I=1,NDIM
      VCORG(I2)=X1(I)+X2(I)*I1
40    I2=I2+1
      I1=I1+1
50    I2=I2+I3
      GO TO 20
C----- CHECK FOR MISSING NODES
60    I1=NNT*NDIM+1
      I2=0
      I3=NNT+1
      DO 90 I=1,NNT
      I1=I1-NDIM
      I3=I3-1
      IF(VCORG(I1)-SPECL) 70,80,70
70    IF(I2.EQ.0) I2=I3
      GO TO 90
80    IF(I2.EQ.0) CALL ERROR(16,I3,I3,0)

```

```

      IF(I2.NE.0) CALL ERROR(17,I3,I3,1)
90    CONTINUE
      IF(I2.NE.NNT) CALL ERROR(18,NNT,I2,0)
C----- TOTAL NUMBER OF D.O.F.
      NDLT=0
      I1=NNT+1
      DO 100 I=2,I1
100    NDLT=NDLT+KDLNC(I)
C----- OUTPUT
      IF(M.LT.2) GO TO 120
      WRITE(MP,2020)
2020  FORMAT(/10X,'NODE D.O.F.',5X,'X',11X,'Y',11X,'Z')
      I1=1
      I2=NDIM
      DO 110 IN=1,NNT
      WRITE(MP,2030) IN,KDLNC(IN+1),(VCORG(I),I=I1,I2)
2030  FORMAT(10X,2I5,3E12.5)
      I1=I1+NDIM
110    I2=I2+NDIM
120    RETURN
      END
      SUBROUTINE BLCOND
C=====
C      TO CALL BLOCK 'COND'
C      TO READ BOUNDARY CONDITIONS AND GENERATE TABLE (NEQ)
C=====
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*4 TBL
      COMMON/COOR/NDIM,NNT,NDLN,NDLT
      COMMON/COND/NCLT,NCLZ,NCLNZ
      COMMON/ALLO/NVA,IVA,IVAMAX,NREEL,NTBL
      COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
      COMMON/LOCA/LCORG,LDLNC,LNEQ,LDIMP,LCODE,LINTF,LPREG,LLD,LLOC,
1     LCORE,LNE,LPRREE,LNPEC,LSDEG,LDLE,LKE,LFE,LKGS,LKGD,LKGJ,LFG,
2     LRES,LDLG,LNELS,LEB,LPB
      COMMON VA(20000)
      DIMENSION TBL(2)
      DATA TBL/'NEQ ','DIMP'/
C-----
      WRITE(*,3000)
3000  FORMAT(' BLCOND')
      WRITE(MP,2000) M
2000  FORMAT('// INPUT OF BOUNDARY CONDITIONS (M=',I2,')// ',1,
1     35('=')/)
      IF(LNEQ.EQ.1) CALL ESPACE(NDLT,0,TBL(1),LNEQ)
      IF(LDIMP.EQ.1) CALL ESPACE(NDLT,1,TBL(2),LDIMP)
      CALL EXCOND(VA(LCORG),VA(LDLNC),VA(LNEQ),VA(LDIMP))
      CALL DSPACE(LDIMP+NCLT,1,TBL(2))
      RETURN
      END
      SUBROUTINE EXCOND(VCORG,KDLNC,KNEQ,VDIMP)
C=====
C      TO EXECUTE BLOCK 'COND'
C      READ BOUNDARY CONDITIONS AND GENERATE TABLE (NEQ)

```

```

C=====
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/COOR/NDIM,NNT,NDLN,NDLT
      COMMON/COND/NCLT,NCLZ,NCLNZ
      COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
      COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
      DIMENSION KV(16),V(10),ICOD(10)
      DIMENSION VCORG(*),KDLNC(*),KNEQ(*),VDIMP(*)
      DATA L7/7/,L8/8/,L16/16/,X1/0.0D0/,X2/0.0D0/,X3/0.0D0/,ZERO/0.0D/
C-----
C----- CUMULATIVE TABLE KDLNC
      DO 10 IN=1,NNT
10    KDLNC(IN+1)=KDLNC(IN)+KDLNC(IN+1)
      I1=NNT+1
      IF(M.GE.2) WRITE(MP,2000) (KDLNC(IN),IN=1,I1)
2000  FORMAT(//' NUMBER OF D.O.F. PRECEDING EACH NODE   (DLNC)'/
     1 (1X,10I10))
C----- INITIALIZE
      NCLT=0
      NCLNZ=0
      NCLZ=0
      IF(M.GE.0) WRITE(MP,2010)
2010  FORMAT(//' BOUNDARY CONDITIONS CARDS')
C----- READ A B.C. GROUP CARD : 10 CODES + PRESCRIBED VAL.
20    READ(MR,1000) ICOD,(V(I),I=1,L7)
1000  FORMAT(10I1,7F10.0)
      IF(M.GE.0) WRITE(MP,2020) ICOD,(V(I),I=1,L7)
2020  FORMAT(' >>>',10I1/10X,7E10.3)
C----- CHECK FOR A BLANK CARD
      J=0
      DO 30 I=1,10
30    J=J+ICOD(I)
      IF(J.EQ.0) GO TO 110
C----- READ ADDITIONAL CARD IF REQUIRED
      I2=0
      DO 40 ID=1,NDLN
      IF(ICOD(ID).LT.2) GO TO 40
      I2=I2+1
      IF(I2.NE.L8) GO TO 40
      READ(MR,1010) (V(I),I=L8,NDLN)
1010  FORMAT(10X,7F10.0)
      IF(M.GE.0) WRITE(MP,2030) (V(I),I=L8,NDLN)
2030  FORMAT(10X,7E10.3)
40    CONTINUE
C----- READ NODE CARDS
50    READ(MR,1020) (KV(IN),IN=1,L16)
1020  FORMAT(16I5)
      IF(M.GE.0) WRITE(MP,2040) (KV(IN),IN=1,L16)
2040  FORMAT(' >>> (NODES)'/16I5)
C----- FORM NEQ
      DO 100 IN=1,L16
      I2=KV(IN)
C----- END OF GROUP OF B.C. OR END OF NODES OR ANALYSIS OF A NODE
      IF(I2) 20,20,60

```

```

60      IF(I2.GT.NNT) CALL ERROR(32,I2,NNT,1)
       I1=KDLNC(I2)
       IDN=KDLNC(I2+1)-I1
C----- GENERATE VDIMP, PUT IT IN KNEQ (THE PRESCRIBED D.O.F. ADDRESS)
       IV=0
       DO 90 ID=1, IDN
          I1=I1+1
          IC=ICOD(ID)-1
          IF(IC) 90,70,80
70      NCLT=NCLT+1
       VDIMP(NCLT)=ZERO
       NCLZ=NCLZ+1
       KNEQ(I1)=-NCLZ
       GO TO 90
80      NCLT=NCLT+1
       IV=IV+1
       VDIMP(NCLT)=V(IV)
       NCLNZ=NCLNZ+1
       KNEQ(I1)=-NCLT
90      CONTINUE
100     CONTINUE
C----- ADDITIONAL CARD OF NODE NUMBERS
       GO TO 50
C----- GENERATE EQUATION NUMBERS IN NEQ
110     I1=0
       DO 150 IN=1, NNT
          ID=KDLNC(IN)
120     ID=ID+1
       IF(ID.GT.KDLNC(IN+1)) GO TO 150
       IF(KNEQ(ID)) 120,130,120
130     I1=I1+1
       KNEQ(ID)=I1
       GO TO 120
150     CONTINUE
       NEQ=I1
C----- OUTPUT
       IF(M.LT.0) GO TO 170
       WRITE(*,2050) NNT,NDLT,NEQ,NCLNZ,NCLZ,NCLT
       WRITE(MP,2050) NNT,NDLT,NEQ,NCLNZ,NCLZ,NCLT
2050     FORMAT(//
1      15X,'TOTAL NUMBER OF NODES'                                (NNT)=' ,I5/
2      15X,'TOTAL NUMBER OF D.O.F.'                               (NDLT)=' ,I5/
3      15X,'NUMBER OF EQUATIONS TO BE SOLVED'                  (NEQ)=' ,I5/
4      15X,'NUMBER OF PRESCRIBED NON ZERO D.O.F.'             (NCLNZ)=' ,I5/
5      15X,'NUMBER OF PRESCRIBED ZERO D.O.F.'                 (NCLZ)=' ,I5/
6      15X,'TOTAL NUMBER OF PRESCRIBED D.O.F.'                (NCLT)=' ,I5/
       IF(M.GE.2.AND.NCLT.GT.0) WRITE(MP,2060)(VDIMP(I),I=1,NCLT)
2060     FORMAT(//' PRESCRIBED VALUES (VDIMP)'//(10X,10E12.5))
       WRITE(MP,2070)
2070     FORMAT(//' NODAL COORDINATES ARRAY'//)
1      ' NO D.L.',5X,'X',12X,'Y',12X,'Z',10X,'EQUATION NUMBER',
2      '(NEQ)'/)
       I2=0
       DO 160 IN=1, NNT

```

```

I1=I2+1
I2=I2+NDIM
ID1=KDLNC(IN)+1
ID2=KDLNC(IN+1)
ID=ID2-ID1+1
IF(ID2.LT.ID1) ID2=ID1
X1=VCORG(I1)
IF(NDIM.GE.2) X2=VCORG(I1+1)
IF(NDIM.GE.3) X3=VCORG(I1+2)
160 WRITE(MP,2080) IN, ID, X1, X2, X3, (KNEQ(I), I=ID1, ID2)
2080 FORMAT(1X,2I5,3E12.5,10X,10I6)
170 RETURN
END
SUBROUTINE BLPREL
C=====
C      TO CALL BLOCK 'PREL'
C      TO READ ELEMENT PROPERTIES (INCLUDING MODEL PARAMETERS)
C=====
IMPLICIT REAL*8(A-H,O-Z)
CHARACTER*4 TBL
COMMON/PREL/NGPE,NPRE,NPRM,NMNL,NHIS
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
COMMON/LOCA/LCORG,KDLNC,LNEQ,LDIMP,LCODE,LINTF,LREG,LLD,LLOC,
1 LCORE,LNE,LPREE,LNPEG,LSDEG,LDLE,LKE,LFE,LKGS,LKGD,LKGI,LFG,
2 LRES,LDLG,LNELS,LEB,LPB
COMMON/TVL3/YWAT,ROWA,BKWA,ATMP
COMMON VA(20000)
DIMENSION TBL(4),IN(3),WA(3)
DATA TBL/'CODE','INTF','PREG','V   ','/,ZERO/0.0/DO/
C-----
3000 WRITE(*,3000)
3000 FORMAT(' BLPREL')
C----- READ NUMBER OF GROUPS AND PROPERTIES PER GROUP
READ(MR,1000) IN,WA
1000 FORMAT(3I5,3F10.0)
IF(IN(1).GT.0) NGPE=IN(1)
IF(IN(2).GT.0) NPRE=IN(2)
IF(IN(3).GT.0) NPRM=IN(3)
IF(WA(1).NE.ZERO) ROWA=WA(1)
IF(WA(2).NE.ZERO) BKWA=WA(2)
IF(WA(3).NE.ZERO) ATMP=WA(3)
WRITE(MP,2000) M,NGPE,NPRE,NPRM
2000 FORMAT(//' INPUT OF ELEMENT PROPERTIES (M=',I2,')'/' ',34('=')/
1 15X,'NUMBER OF GROUPS OF PROPERTIES          (NGPE=)',I5/
2 15X,'MAX. NUMBER OF BASIC PROPERTIES PER GROUP (NPRE=)',I5/
3 15X,'MAX. NUMBER OF MODEL PROPERTIES PER GROUP (NPRM=)',I5)
IF(LCODE.EQ.1) CALL ESPACE(NGPE,0,TBL(1),LCODE)
IF(LINTF.EQ.1) CALL ESPACE(NGPE,0,TBL(2),LINTF)
IF(LREG.EQ.1) CALL ESPACE(NGPE*(NPRE+NPRM),1,TBL(3),LREG)
CALL ESPACE(NPRE+NPRM,1,TBL(4),L1)
CALL EXPREL(VA(LCODE),VA(LINTF),VA(LREG),VA(L1))
CALL DSPACE(L1,1,TBL(4))
RETURN
END

```

```

SUBROUTINE EXPREL(KCODE,KINTF,VPREG,V1)
C=====
C      TO EXECUTE BLOCK 'PREL'
C      READ ELEMENT PROPERTIES (INCLUDING MODEL PARAMETERS)
C=====
IMPLICIT REAL*8(A-H,O-Z)
COMMON/PREL/NGPE,NPRE,NPRM,NMNL,NHIS
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
DIMENSION KCODE(*),KINTF(*),VPREG(*),V1(*)
C-----
IF(M.EQ.0) WRITE(MP,2000)
2000 FORMAT(//' CARDS OF ELEMENT PROPERTIES')
C----- READ A GROUP
NPRT=NPRE+NPRM
J=1
K=1
10 I1=MINO(7,NPRT)
READ(MR,1000) IGPE,MODEL,IFEL,(V1(I),I=1,I1)
1000 FORMAT(I5,I3,I2,7F10.0)
C----- CHECK IF DEAL WITH ANY NONLINEAR MODEL
IF(MODEL.GT.0) NMNL=1
C----- CHECK HOW MANY HISTORY DATA AT A G.P.
IF(MODEL.EQ.4) NHIS=2
IF(M.EQ.0) WRITE(MP,2010) IGPE,MODEL,IFEL,(V1(I),I=1,I1)
2010 FORMAT(' >>>'/I5,I3,I2,7E10.3)
IF(IGPE.LE.0) GO TO 40
IF(IGPE.GT.NGPE) CALL ERROR(41,IGPE,NGPE,1)
KCODE(K)=MODEL
KINTF(K)=IFEL
IF(NPRT.LE.7) GO TO 20
C----- READ MORE PROPERTIES
N=1
I1=8
15 N=N+1
I2=MINO(N*7,NPRT)
READ(MR,1010) (V1(I),I=I1,I2)
1010 FORMAT(10X,7F10.0)
IF(M.GE.0) WRITE(MP,2020) (V1(I),I=I1,I2)
2020 FORMAT(10X,7E10.3)
I1=I2+1
IF(NPRT.GT.I2) GO TO 15
20 DO 30 I=1,NPRT
VPREG(J)=V1(I)
30 J=J+1
K=K+1
GO TO 10
40 RETURN
END
SUBROUTINE BLELEM
C=====
C      TO CALL BLOCK 'ELEM'
C      TO READ ELEMENT DATA
C=====
IMPLICIT REAL*8(A-H,O-Z)

```

```

CHARACTER*4 TBL
COMMON/COOR/NDIM,NNT,NDLN,NDLT
COMMON/PREL/NGPE,NPRE,NPRM,NML,NHIS
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1 NPEM,NELS
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDL
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
COMMON/LOCA/LCORG,LDLNC,LNEQ,LDIMP,LCODE,LINFT,LPREG,LLD,LLOCE,
1 LCORE,LNE,LPREE,LNPEG,LSDEG,LDLE,LKE,LFE,LKGS,LKGD,LKGI,LFG,
2 LRES,LDLG,LNELS,LEB,LPB
COMMON VA(20000)
DIMENSION TBL(7),IN(8)
DATA TBL/'LD ','LOCE','CORE','NPEG','SDEG','NE ','PREE'/
C-----
C----- WRITE(*,3000)
3000 FORMAT(' BLELEM')
C----- OPEN ELEMENT PROPERTY FILE ME (DIRECT ACCESS)
C----- DETERMINE THE BLOCK LENGTH
LR=7*4+NNEL*NDLN*4+NNEL*NDIM*8+NNEL*4
OPEN(ME,FILE='ME.DAT',STATUS='NEW',ACCESS='DIRECT',RECL=LR)
READ(MR,1000) IN
1000 FORMAT(8I5)
IF(IN(1).GT.0) NELT=IN(1)
IF(IN(2).GT.0) NNEL=IN(2)
IF(IN(3).GT.0) NKEL=IN(3)
IF(IN(4).GT.0) NTPE=IN(4)
IF(IN(5).GT.0) NSDG=IN(5)
IF(IN(6).GT.0) NLAG=IN(6)
IF(IN(7).NE.0) NSYM=1
IF(IN(8).NE.0) NIDENT=1
WRITE(MP,2000) M,NELT,NNEL,NKEL,NTPE,NSDG,NSYM,NIDENT
2000 FORMAT(// INPUT OF ELEMENTS (M=',I2,')'/' ',24('=')/
1 15X,'MAX. NUMBER OF ELEMENTS          (NELT)=',I5/
2 15X,'MAX. NUMBER OF NODES PER ELEMENT (NNEL)=',I5/
3 15X,'DEFAULT ELEMENT CASE           (NKEL)=',I5/
4 15X,'DEFAULT ELEMENT TYPE          (NTPE)=',I5/
5 15X,'ELEMENT BEHAVIOR CODE         (NSDG)=',I5/
6 15X,'INDEX FOR NON SYMMETRIC PROBLEM (NSYM)=',I5/
7 15X,'INDEX FOR IDENTICAL ELEMENTS   (NIDENT)=',I5)
IF(LLD.EQ.1) CALL ESPACE(NEQ+1,0,TBL(1),LLD)
IF(LLOCE.EQ.1) CALL ESPACE(NNEL*NDLN,0,TBL(2),LLOCE)
IF(LCORE.EQ.1) CALL ESPACE(NNEL*NDIM,1,TBL(3),LCORE)
IF(LNPEG.EQ.1) CALL ESPACE(NELT,0,TBL(4),LNPEG)
IF(NSDG.GT.0.AND.LSDEG.EQ.1) CALL ESPACE(NELT,0,TBL(5),LSDEG)
IF(LNE.EQ.1) CALL ESPACE(NNEL,0,TBL(6),LNE)
IF(NPRE.GT.0.AND.LPREE.EQ.1) CALL ESPACE(NPRE+NPRM,1,TBL(7),LPREE)
CALL EXELEM(VA(LCORG),VA(LDLNC),VA(LPREG),VA(LLOCE),VA(LCORE),
1 VA(LNPEG),VA(LSDEG),VA(LNE),VA(LNEQ),VA(LLD))
WRITE(MP,2010) NKG,NPG
2010 FORMAT(15X,'LENGTH OF A TRIANGLE IN KG          (NKG)=',I10/
1 15X,'NUMBER OF INTEGRATION POINTS      (NPG)=',I10/)
RETURN
END
SUBROUTINE EXELEM(VCORG,KDLNC,VPREG,KLOCE,VCORE,KNPEG,KSDEG,KNE,

```

```

1 KNEQ,KLD)
C=====
C      TO EXECUTE BLOCK 'ELEM'
C      READ ELEMENT DATA
C=====
IMPLICIT REAL*8(A-H,O-Z)
COMMON/COOR/NDIM,NNT,NDLN,NDLT
COMMON/PREL/NGPE,NPRE,NPRM,NMNL,NHIS
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1 NPEM,NELS
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
COMMON/RGDT/IEL,IKEL,ITPE,ITPE1,ISDE,IMODEL,IFEL,IDLE,ICE,INEL,
1 IPG,ICODE,IMATD,INSE,INPE,IDL0,INEL0,IPGO,IRES
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
DIMENSION VCORG(*),KDLNC(*),VPREG(*),KLOCE(*),VCORE(*),KNPEG(*),
1 KSDEG(*),KNE(*),KNEQ(*),KLD(*)
DATA I9/9/,I15/15/
C-----
C----- INITIALIZE
NSEM=0
NPEM=0
NDLE=0
IELT=0
NPG=0
IF(M.GT.0) WRITE(MP,2000)
2000 FORMAT(//' ELEMENTS CARDS')
C----- READ AN ELEMENT CARD
10 READ(MR,1000) IEL,IGEN,INCR,IKEL,ITPE,IGPE,ISDE,
1           (KNE(IN),IN=1,I9)
1000 FORMAT(16I5)
IF(M.GT.0) WRITE(MP,2010) IEL,IGEN,INCR,IKEL,ITPE,IGPE,ISDE,
1           (KNE(IN),IN=1,I9)
2010 FORMAT(' >>>',16I5)
IF(IEL) 80,80,20
C----- NUMBER OF NODES AND ADDITIONAL CARDS AS REQUIRED
20 INEL=0
I1=1
I2=I9
30 DO 40 IN=I1,I2
IF(KNE(IN).EQ.0) GO TO 45
INEL=INEL+1
40 CONTINUE
I1=I2+1
I2=I1+I5
READ(MR,1000) (KNE(IN),IN=I1,I2)
IF(M.GT.0) WRITE(MP,2010) (KNE(IN),IN=I1,I2)
GO TO 30
C----- CHECKING
45 IF(INEL.GT.NNEL) CALL ERROR(51,INEL,NNEL,1)
IF(INCR.EQ.0) INCR=1
IF(IKEL.EQ.0) IKEL=NKEL
IF(ITPE.EQ.0) ITPE=NTPE
IF(IGPE.GT.NGPE) CALL ERROR(53,IGPE,NGPE,1)
IF(ISDE.GT.NSDG) CALL ERROR(54,ISDE,NSDG,1)

```

```

C----- ELEMENT GENERATION
  IF(IGEN.EQ.0) IGEN=1
  DO 70 IE=1,IGEN
    IF(IEL.GT.NELT) CALL ERROR(55,IEL,NELT,1)
C----- GENERATE KLOCE AND UPDATE KLD
  CALL LOCELK(KDLNC,KNE,KNEQ,KLOCE,KLD)
C----- GENERATE ELEMENT COORDINATES
  ICE=0
  DO 50 I1=1,INEL
    IC=(KNE(I1)-1)*NDIM
    DO 50 I2=1,NDIM
      ICE=ICE+1
      IC=IC+1
      VCORE(ICE)=VCORG(IC)
50  CONTINUE
C----- GENERATE GLOBAL ELEMENT PROPERTY AND BEHAVIOR TABLE
  KNPEG(IEL)=IGPE
  IF(NSDG.GT.0) KSDEG(IEL)=ISDE
C----- UPDATE TOTAL NUMBER OF ELEMENTS DELETED
  IF(IGPE.EQ.0) NELS=NELS+1
C----- CHECK ELEMENT NODE NUMBERS AND D.O.F.
  IPGO=0
  ICODE=1
  CALL ELEMLB(VCORE,VPREE,VDLE,VKE,VFE)
  IF(INEL.EQ.INELO.AND.IDLE.EQ.IDLE0) GO TO 55
  WRITE(MP,2020) IEL,INEL,INELO,IDL0,IDL0
2020 FORMAT(' ** ELEMENT',I5,' INCONSISTENT'/5X,'INEL=',I4,' INELO=',I5
1/ 5X,'IDLE=',I5,' IDLE0=',I5)
C----- UPDATE TOTAL NUMBER OF INTEGRATION POINTS
55  NPG=NPG+IPGO
C----- CALCULATE TOTAL NUMBER OF STRESSES AT INTEGRATION POINTS
  INSE=IMATD*IPGO
  INPE=0
  IF(NSDG.GT.0) INPE=IPGO
C----- FIND MAX. NSEL AND INPE
  NSEL=2*INSE+INPE
  IF(NSEM.LT.NSEL) NSEM=NSEL
  IF(NPEM.LT.IPGO) NPEM=IPGO
C----- STORE ON ELEMENT FILE
  CALL WRELEM(ME,KLOCE,VCORE,KNE)
  IELT=IELT+1
C----- PRINT ELEMENT CHARACTERISTICS
  CALL PRELEM(KLOCE,VCORE,KNE,KNPEG)
C----- NEXT ELEMENT TO BE GENERATED OR READ
  DO 60 IN=1,INEL
60  KNE(IN)=KNE(IN)+INCR
  IF(IDLE.GT.NDLE) NDLE=IDLE
70  IEL=IEL+1
  GO TO 10
C----- CHECK IF TOTAL NUMBER OF ELEMENT IS EXCEEDED
80  IF(IEL.NE.NELT) CALL ERROR(57,IEL,NELT,1)
C----- PRINT BAND HEIGHTS
  IMA=0
  IMO=0

```

```

I1=NEQ+1
DO 90 I=2,I1
J=KLD(I)
IF(J.GT.IMA)IMA=J
90 IMO=IMO+J
C=IMO
C=C/NEQ
WRITE(MP,2030) C,IMA
2030 FORMAT(15X,'MEAN BAND HEIGHT=',F8.1,' MAXIMUM=',I5)
IF(M.GE.2) WRITE(MP,2040) (KLD(I),I=1,I1)
2040 FORMAT(//' TABLE OF BAND HEIGHTS'/(10X,20I5))
C----- TRANSFORM KLD INTO ADDRESSES OF COLUMN TOP TERM
IF(NSYM.EQ.0) NKE=(NDLE*(NDLE+1))/2
IF(NSYM.EQ.1) NKE=NDLE*NDLE
KLD(1)=1
DO 100 ID=2,I1
100 KLD(ID)=KLD(ID-1)+KLD(ID)
NKG=KLD(I1)-1
IF(M.GE.2) WRITE(MP,2050) (KLD(ID),ID=1,I1)
2050 FORMAT(//' TABLE OF ADDRESSES OF COLUMN TOP TERMS (LD)'/
1           (10X,20I6))
C----- NUMBER OF STRESSES AND NUMBER OF PWP PER ELEMENT(CUMULATIVE)
RETURN
END
SUBROUTINE LOCELD(KDLNC,KNE,KNEQ,KLOCE,KLD)
C=====
C      TO FORM THE ELEMENT LOCALIZATION TABLE (LOCE)
C      AND UPDATE COLUMN HEIGHTS FOR A GIVEN ELEMENT
C=====
COMMON/COOR/NDIM,NNT,NDLN,NDLT
COMMON/RGDTIEL,IKEL,ITPE,ITPE1,ISDE,IMODEL,IFEL,IDLE,ICE,INEL,
1  IPG,ICODE,IMATD,INSE,INPE,IDELO,INEL0,IPGO,IRES
DIMENSION KDLNC(*),KNE(*),KNEQ(*),KLOCE(*),KLD(*)
DATA NDLMAX/32000/
C-----
C----- GENERATE KLOCE FROM KNEQ
IDLE=0
LOCMIN=NDLMAX
DO 20 IN=1,INEL
INN=KNE(IN)
IF(INN.GT.NNT) CALL ERROR(56,INN,NNT,1)
IEQ=KDLNC(INN)
IEQ1=KDLNC(INN+1)
10 IF(IEQ.GE.IEQ1) GO TO 20
IEQ=IEQ+1
IDLE=IDLE+1
J=KNEQ(IEQ)
KLOCE(IDLE)=J
IF(J.LT.LOCMIN.AND.J.GT.0) LOCMIN=J
GO TO 10
20 CONTINUE
C----- UPDATE TABLE OF COLUMN HEIGHTS (KLD)
DO 30 ID=1,IDL
J=KLOCE(ID)

```

```

IF(J.LE.0) GO TO 30
IH=J-LOCMIN
IF(IH.GT.KLD(J+1))KLD(J+1)=IH
30  CONTINUE
RETURN
END
SUBROUTINE PRELEM(KLOCE,VCORE,KNE,KNPEG)
C=====
C   PRINT DATA DEFINING AN ELEMENT
C=====
IMPLICIT REAL*8(A-H,O-Z)
COMMON/RGDTIEL,IKEL,ITPE,ITPE1,ISDE,IMODEL,IFEL,IDLE,ICE,INEL,
1 IPG,ICODE,IMATD,INSE,INPE,IDLEO,INELO,IPGO,IRES
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
DIMENSION KLOCE(*),VCORE(*),KNE(*),KNPEG(*)
C-----
2000 IF(M.GE.0) WRITE(MP,2000) IEL,IKEL,ITPE,INEL,KNPEG(IEL),ISDE
      FORMAT(10X,'ELEMENT:',I5,' KIND:',I2,' TYPE:',I2,' N.P.:',I2,
      1 ' D.O.F.:',I3,' PR #:',I2,' S.D.:',I2)
      IF(M.GE.0) WRITE(MP,2010) (KNE(I),I=1,INEL)
2010 FORMAT(15X,'CONNECTIVITY (NE)',20I5/(32X,20I5))
      IF(M.LT.1) GO TO 10
      WRITE(MP,2020) (KLOCE(I),I=1,IDLE)
2020 FORMAT(15X,'LOCALIZATN (LOCE)',20I5/(32X,20I5))
      WRITE(MP,2030) (VCORE(I),I=1,ICE)
2030 FORMAT(15X,'COORDINATES(CORE)',8E12.5/(32X,8E12.5))
10  RETURN
END
SUBROUTINE WRELEM(ME,KLOCE,VCORE,KNE)
C=====
C   WRITE ELEMENT PROPERTIES ON RAMDON ACCESS FILE ME
C=====
IMPLICIT REAL*8(A-H,O-Z)
COMMON/RGDTIEL,IKEL,ITPE,ITPE1,ISDE,IMODEL,IFEL,IDLE,ICE,INEL,
1 IPG,ICODE,IMATD,INSE,INPE,IDLEO,INELO,IPGO,IRES
DIMENSION KLOCE(*),VCORE(*),KNE(*)
C-----
1 (KLOCE(I),I=1,INEL),(VCORE(I),I=1,ICE),(KNE(I),I=1,INEL)
RETURN
END
SUBROUTINE RDELEM(ME,KLOCE,VCORE,KNE)
C=====
C   READ ELEMENT PROPERTIES FROM RAMDON ACCESS FILE ME
C=====
IMPLICIT REAL*8(A-H,O-Z)
COMMON/RGDTIEL,IKEL,ITPE,ITPE1,ISDE,IMODEL,IFEL,IDLE,ICE,INEL,
1 IPG,ICODE,IMATD,INSE,INPE,IDLEO,INELO,IPGO,IRES
DIMENSION KLOCE(*),VCORE(*),KNE(*)
C-----
1 (KLOCE(I),I=1,INEL),(VCORE(I),I=1,ICE),(KNE(I),I=1,INEL)
RETURN
END

```

```

SUBROUTINE BLCONC
C=====
C   TO CALL BLOCK 'CONC'
C   TO READ CONCENTRATED LOADS
C=====
IMPLICIT REAL*8(A-H,O-Z)
CHARACTER*4 TBL
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
COMMON/LOCA/LCORG,LDLNC,LNEQ,LDIMP,LCODE,LINTF,LPREG,LLD,LLOCE,
1 LCORE,LNE,LPREE,LNPEG,LSDEG,Ldle,LKE,LFE,LKGS,LKGD,LKGI,LFG,
2 LRES,LDLG,LNELS,LEB,LPB
COMMON VA(2000)
DATA TBL/'FG' /
C-----
      WRITE(*,3000)
3000 FORMAT(' BLCONC')
      WRITE(MP,2000) M
2000 FORMAT(//' INPUT OF CONCENTRATED LOADS (M=',I2,')'//',
1 39('=')
      IF(LFG.EQ.1) CALL ESPACE(NEQ,1,TBL,LFG)
      CALL EXCONC(VA(LFG),VA(LDLNC),VA(LNEQ))
      RETURN
      END
      SUBROUTINE EXCONC(VFG,KDLNC,KNEQ)
C=====
C   TO EXECUTE BLOCK 'CONC'
C   READ CONCENTRATED LOADS
C=====
IMPLICIT REAL*8(A-H,O-Z)
COMMON/COOR/NDIM,NNT,NDLN,NDLT
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1 NPEM,NELS
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
DIMENSION KV(16),V(14)
DIMENSION VFG(*),KDLNC(*),KNEQ(*)
DATA L16/16/,ZERO/0.D0/
C-----
C----- READ DATA
      IF(M.GE.0) WRITE(MP,2000)
2000 FORMAT(//' CARDS OF NODAL LOADS'//)
      IO=MINO(7,NDLN)
10     READ(MR,1000) IG,(V(I),I=1,IO)
1000  FORMAT(I5,7F10.0)
      IF(NDLN.GT.7) READ(MR,1005) (V(I),I=8,NDLN)
1005  FORMAT(5X,7F10.0)
      IF(M.GE.0) WRITE(MP,2010) IG,(V(I),I=1,NDLN)
2010  FORMAT(' >>>',I5,7E12.5/(' >>>',5X,7E12.5))
      IF(IG.LE.0) GO TO 60
20     READ(MR,1010) (KV(I),I=1,L16)
1010  FORMAT(16I5)
      IF(M.GE.0) WRITE(MP,2020) (KV(I),I=1,L16)
2020  FORMAT(' >>>',16I5)

```

```

C----- DECODE NODAL DATA
      DO 50 IN=1,L16
      I1=KV(IN)
      IF(I1.GT.NNT) CALL ERROR(61,I1,NNT,1)
      IF(I1) 10,10,30
 30     ID1=KDLNC(I1)+1
      ID2=KDLNC(I1+1)
      J=0
      DO 50 ID=ID1,ID2
      J=J+1
      IEQ=KNEQ(ID)
      IF(IEQ) 50,50,40
 40     VFG(IEQ)=VFG(IEQ)+V(J)
 50     CONTINUE
      GO TO 20
C----- OUTPUT
 60     IF(M.GE.1) WRITE(MP,2030) (VFG(I),I=1,NEQ)
2030   FORMAT(//' TOTAL LOAD VECTOR'/(10X,10E12.5))
      RETURN
      END
      SUBROUTINE BLBODY
C=====
C      TO CALL BLOCK 'BODY'
C      TO ASSEMBLE DISTRIBUTED LOADS (ELEMENT FUNCTION 7)
C=====

      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*4 TBL
      COMMON/COOR/NDIM,NNT,NDLN,NDLT
      COMMON/ELEM/NELT,NNEL,NKEL,NTP,E,NSDG,LAG,NIDENT,NPG,NSEM,
      1 NPEM,NELS
      COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
      COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
      COMMON/LOCA/LCORG,LDLNC,LNEQ,LDIMP,LCODE,LINTF,LPREG,LLD,LLOCE,
      1 LCORE,LNE,LPREE,LNPEC,LSDEG,LDLE,LKE,LFE,LKGS,LKGD,LKGI,LFG,
      2 LRRES,LDLG,LNELS,LEB,LPB
      COMMON VA(20000)
      DIMENSION TBL(8)
      DATA TBL/'FG ','KE ','FE ','DLE ','KGS ','KGD ','KGI ',
      1 'RES '/

C----- ASSEMBLING OF DISTRIBUTED LOADS (M=,I2,')'/1X,40('=')/
 3000  FORMAT(' BLBODY')
      WRITE(MP,2000) M
 2000  FORMAT(//' ASSEMBLING OF DISTRIBUTED LOADS (M=',I2,')'/
      1 1X,40('=')/)
      IF(LFG.EQ.1) CALL ESPACE(NEQ,1,TBL(1),LFG)
      IF(LFE.EQ.1) CALL ESPACE(NDLE,1,TBL(3),LFE)
      IF(LDLE.EQ.1) CALL ESPACE(NDLE,1,TBL(4),LDLE)
      IF(LRRES.EQ.1) CALL ESPACE(NDLT+NEQ,1,TBL(8),LRRES)
      CALL EXBODY(VA(LLD),VA(LLOCE),VA(LCORE),VA(LNPEC),VA(LPREE),
      1 VA(LNE),VA(LKE),VA(LFE),VA(LKGS),VA(LKGD),VA(LKGI),VA(LFG),
      2 VA(LPREG),VA(LSDEG),VA(LDLE))
      RETURN
      END

```

```

SUBROUTINE EXBODY(KLD,KLOCE,VCORE,KNPEG,VPREE,KNE,VKE,VFE,VKGS,
1 VKGD,VKG,I,VFG,VPREG,KSDEG,VDLE)
C=====
C TO EXECUTE BLOCK 'BODY'
C ASSEMBLE DISTRIBUTED LOADS (ELEMENT FUNCTION 7)
C=====
IMPLICIT REAL*8(A-H,O-Z)
COMMON/COOR/NDIM,NNT,NDLN,NDLT
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,LAG,NIDENT,NPG,NSEM,
1 NPEM,NELS
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
DIMENSION KLD(*),KLOCE(*),VCORE(*),KNPEG(*),VPREE(*),KNE(*),
1 VKE(*),VFE(*),VKGS(*),VKGD(*),VKG(I),VFG(*),VPREG(*),KSDEG(*),
2 VDLE(*)
C-----
C----- ASSEMBLE FG
CALL ASFG(KLD,KLOCE,VCORE,KNPEG,VPREE,KNE,VKE,VFE,VKGS,VKGD,VKG,I,
1 VFG,VPREG,KSDEG,VDLE)
C----- OUTPUT
IF(M.EQ.1) WRITE(MP,2000) (VFG(I),I=1,NEQ)
2000 FORMAT(/' GLOBAL LOAD VECTOR (FG)'/(1X,10E12.5))
RETURN
END
SUBROUTINE ASFG(KLD,KLOCE,VCORE,KNPEG,VPREE,KNE,VKE,VFE,VKGS,
1 VKGD,VKG,I,VFG,VPREG,KSDEG,VDLE)
C=====
C ASSEMBLING DISTRIBUTED LOADS IN FG
C=====
IMPLICIT REAL*8(A-H,O-Z)
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,LAG,NIDENT,NPG,NSEM,
1 NPEM,NELS
COMMON/PREL/NGPE,NPRE,NPRM,NMNL,NHIS
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
COMMON/RGDTIEL,IKEL,ITPE,ITPE1,ISDE,IMODEL,IFEL,IDLE,ICE,INEL,
1 IPIG,ICODE,IMATD,INSE,INPE,IDELO,INEL0,IPGO,IRES
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
DIMENSION KLD(*),KLOCE(*),VCORE(*),KNPEG(*),VPREE(*),KNE(*),
1 VKE(*),VFE(*),VKGS(*),VKGD(*),VKG(I),VFG(*),VPREG(*),KSDEG(*),
2 VDLE(*)
C-----
C----- LOOP OVER THE ELEMENTS
DO 20 IEL=1,NELT
C----- READ AN ELEMENT FROM FILE M2
CALL RDELEM(ME,KLOCE,VCORE,KNE)
WRITE(*,3000) IEL
3000 FORMAT(' ELEMENT LOAD VECTOR',I5)
C----- GENERATE ELEMENT PROPERTIES
IC=(KNPEG(IEL)-1)*(NPRE+NPRM)
DO 10 I=1,NPRE+NPRM
10 VPREE(I)=VPREG(IC+I)
IF(NSDG.GT.0) ISDE=KSDEG(IEL)
C----- EVALUATE INTERPOLATION FUNCTIONS IF REQUIRED
IF(ITPE.EQ.ITPE1) GO TO 15

```

```

    ICODE=2
    CALL ELEMLB(VCORE,VPREE,VDLE,VKE,VFE)
C----- EVALUATE ELEMENT VECTOR
15   ICODE=4
    CALL ELEMLB(VCORE,VPREE,VDLE,VKE,VFE)
C----- PRINT ELEMENT VECTOR VFE
    IF(M.GE.2) WRITE(MP,2000) IEL,(VFE(I),I=1,IDLE)
2000 FORMAT('!' VECTOR (FE) , ELEMENT:',I5/(10X,10E12.5))
C----- ASSEMBLE
    CALL ASSEL(0,1,IDLE,NSYM,KLOCE,KLD,VKE,VFE,VKGS,VKGD,VKGI,VFG)
20   ITPE1=ITPE
    RETURN
    END
    SUBROUTINE BLINSI
C=====
C      TO CALL BLOCK 'INSI'  (INSITU MATERIAL LOADS)
C      TO ASSEMBLE EQUIVALENT NODAL LOADS DUE TO INSITU LOADING
C=====
    IMPLICIT REAL*8(A-H,O-Z)
    CHARACTER*4 TBL
    COMMON/COOR/NDIM,NNT,NDLN,NDLT
    COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,LAG,NIDENT,NPG,NSEM,
    1 NPEM,NELS
    COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
    COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
    COMMON/LOCA/LCORE,LNPEC,LNPEG,LNSEQ,LDE,LINTF,LREG,LLD,LLOCE,
    1 LCORE,LNE,LPREE,LNPEC,LNSEQ,LDE,LINTF,LREG,LLD,LLOCE,
    2 LRES,LDLG,LNELS,LEB,LPB
    COMMON/SEQU/NSEQ,NSEQO,ISEQ,IASSEL,IBAR
    COMMON VA(20000)
    DIMENSION TBL(8)
    DATA TBL/'FG ','KE ','FE ','DLE ','KGS ','KGD ','KGI ',
    1 'RES '/
C-----
3000  WRITE(*,3000)
    FORMAT(' BLINSI')
    ISEQ=1
    IASSEL=1
    IBAR=0
    WRITE(MP,2000) M
2000  FORMAT('!' ASSEMBLING OF INSITU LOADS (M=',I2,')!')
    1 1X,40('='))
    IF(LFG.EQ.1) CALL ESPACE(NEQ,1,TBL(1),LFG)
    IF(LFE.EQ.1) CALL ESPACE(NDLE,1,TBL(3),LFE)
    IF(LDLE.EQ.1) CALL ESPACE(NDLE,1,TBL(4),LDLE)
    IF(LRES.EQ.1) CALL ESPACE(NDLT+NEQ,1,TBL(8),LRES)
    CALL EXINSI(VA(LLD),VA(LLOCE),VA(LCORE),VA(LNPEC),VA(LPREE),
    1 VA(LNE),VA(LKE),VA(LFE),VA(LKGS),VA(LKGD),VA(LKGI),VA(LFG),
    2 VA(LNEQ),VA(LDLC),VA(LREG),VA(LSDEG),VA(LDLE))
    RETURN
    END
    SUBROUTINE EXINSI(KLD,KLOCE,VCORE,KNPEG,VPREE,KNE,VKE,VFE,VKGS,
    1 VKGD,VKGI,VFG,KNEQ,KDLC,VPREG,KSDEG,VDLE)
C=====

```

```

C      TO EXECUTE BLOCK 'INSI'
C      ASSEMBLE EQUIVALENT NODAL LOADS (ELEMENT FUNCTION 4)
C=====
IMPLICIT REAL*8(A-H,O-Z)
COMMON/COOR/NDIM,NNT,NDLN,NDLT
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1  NPEM,NELS
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDL
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
DIMENSION KLD(*),KLOCE(*),VCORE(*),KNPEG(*),VPREE(*),KNE(*),
1  VKE(*),VFE(*),VKGS(*),VKGD(*),VKGI(*),VFG(*),KNEQ(*),KDLNC(*),
2  VPREG(*),KSDEG(*),VDLE(*)
C-----
C----- ASSEMBLE FG
CALL ASFG(KLD,KLOCE,VCORE,KNPEG,VPREE,KNE,VKE,VFE,VKGS,VKGD,
1  VKGI,VFG,VPREG,KSDEG,VDLE)
C----- OUTPUT
IF(M.EQ.1) WRITE(MP,2000) (VFG(I),I=1,NEQ)
2000 FORMAT(/' GLOBAL LOAD VECTOR (FG)'/(1X,10E12.5))
RETURN
END
SUBROUTINE BLLOAD
C=====
C      TO CALL BLOCK 'LOAD'
C      TO READ CONCENTRATED LOADS (CONSTRUCTION SEQUENCES)
C=====
IMPLICIT REAL*8(A-H,O-Z)
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
COMMON/LOCA/LCORG,LDLNC,LNEQ,LDIMP,LCODE,LINTF,LPREG,LLD,LLOCE,
1  LCORE,LNE,LFREE,LNPEG,LSDEG,LDLE,LKE,LFE,LKGS,LKGD,LKGI,LFG,
2  LRES,LDLG,LNELS,LEB,LPB
COMMON/SEQU/NSEQ,NSEQO,ISEQ,IASSEL,IBAR
COMMON VA(20000)
C-----
WRITE(*,3000)
3000 FORMAT(' BLLOAD')
ISEQ=7
IASSEL=0
IF(IBAR.EQ.1) IASSEL=1
IBAR=0
WRITE(MP,2000) M
2000 FORMAT(//' INPUT OF CONCENTRATED LOADS (M=',I2,')'//',
1  39('!'))
CALL EXLOAD(VA(LFG),VA(LDLNC),VA(LNEQ))
RETURN
END
SUBROUTINE EXLOAD(VFG,KDLNC,KNEQ)
C=====
C      TO EXECUTE BLOCK 'LOAD'
C      READ CONCENTRATED LOADS
C=====
IMPLICIT REAL*8(A-H,O-Z)
COMMON/COOR/NDIM,NNT,NDLN,NDLT
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,

```

```

1 NPEM,NELS
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
DIMENSION KV(16),V(14)
DIMENSION VFG(*),KDLNC(*),KNEQ(*)
DATA L16/16/,ZERO/0.D0/
C-----
C----- INITIALIZE VFG
DO 100 I=1,NEQ
100 VFG(I)=ZERO
C----- READ DATA
IF(M.GE.0) WRITE(MP,2000)
2000 FORMAT('' CARDS OF NODAL LOADS'')
IO=MIN0(7,NDLN)
10 READ(MR,1000) IG,(V(I),I=1,IO)
1000 FORMAT(15,7F10.0)
IF(NDLN.GT.7) READ(MR,1005) (V(I),I=8,NDLN)
1005 FORMAT(5X,7F10.0)
IF(M.GE.0) WRITE(MP,2010) IG,(V(I),I=1,NDLN)
2010 FORMAT(' >>>',I5,7E12.5/(' >>>',5X,7E12.5))
IF(IG.LE.0) GO TO 60
20 READ(MR,1010) (KV(I),I=1,L16)
1010 FORMAT(16I5)
IF(M.GE.0) WRITE(MP,2020) (KV(I),I=1,L16)
2020 FORMAT(' >>>',16I5)
C----- DECODE NODAL DATA
DO 50 IN=1,L16
I1=KV(IN)
IF(I1.GT.NNT) CALL ERROR(61,I1,NNT,1)
IF(I1) 10,10,30
30 ID1=KDLNC(I1)+1
ID2=KDLNC(I1+1)
J=0
DO 50 ID=ID1,ID2
J=J+1
IEQ=KNEQ(ID)
IF(IEQ) 50,50,40
40 VFG(IEQ)=VFG(IEQ)+V(J)
50 CONTINUE
GO TO 20
C----- OUTPUT
60 IF(M.GE.1) WRITE(MP,2030) (VFG(I),I=1,NEQ)
2030 FORMAT('' TOTAL LOAD VECTOR'/(10X,10E12.5))
RETURN
END
SUBROUTINE BLDWAT
C=====
C TO CALL BLOCK 'DWAT' (LOWER WATER LEVEL)
C TO ASSEMBLE EQUIVALENT NODAL LOADS DUE TO DEWATERING
C=====
IMPLICIT REAL*8(A-H,O-Z)
CHARACTER*4 TBL
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1 NPEM,NELS

```

```

COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
COMMON/LOCA/LCORG,LDLNC,LNEQ,LDIMP,LCODE,LINTF,LPREG,LLD,LLOCE,
1 LCORE,LNE,LPREE,LNPEG,LSDEG,LDLE,LKE,LFE,LKGS,LKGD,LKGI,LFG,
2 LRES,LDLG,LNELS,LEE,LPB
COMMON/SEQU/NSEQ,NSEQO,ISEQ,IASSEL,IBAR
COMMON VA(20000)
DATA TBL/'NELS'/
C-----
      WRITE(*,3000)
3000 FORMAT(' BLDWAT')
      ISEQ=2
      IASSEL=0
      IBAR=0
      READ(MR,1000) NELS
1000 FORMAT(I5)
      WRITE(MP,2000) M,NELS
2000 FORMAT(// ASSEMBLING OF DEWATER LOADS (M=',I2,')/1X,40('=')
1 15X,'NUMBER OF DWAT ELEMENTS' (NELS)=' ,I5/')
      IF(NELS.EQ.0) WRITE(MP,2010)
2010 FORMAT(// ** ERROR, NO ELEMENT FOR (DWAT) '/')
      IF(NELS.EQ.0) STOP
      CALL ESPACE(NELS,0,TBL,LNELS)
      CALL EXDWAT(VA(LLD),VA(LDIMP),VA(LLOCE),VA(LCORE),VA(LNPEG),
1  VA(LPREE),VA(LNE),VA(LKE),VA(LFE),VA(LKGS),VA(LKGD),VA(LKGI),
2  VA(LFG),VA(LNEQ),VA(LDLNC),VA(LPREG),VA(LSDEG),VA(LDLE),
3  VA(LDLG),VA(LNELS))
      CALL DSPACE(LNELS,0,TBL)
      RETURN
      END
      SUBROUTINE EXDWAT(KLD,VDIMP,KLOCE,VCORE,KNPEG,VPREE,KNE,VKE,VFE,
1  VKGS,VKGD,VKGI,VFG,KNEQ,KDLNC,VPREG,KSDEG,VDLE,VDLG,KNELS)
C=====
C TO EXECUTE BLOCK 'DEWA'
C ASSEMBLE EQUIVALENT NODAL LOADS (ELEMENT FUNCTION 6)
C=====
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/COOR/NDIM,NNT,NDLN,NDLT
      COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1  NPEM,NELS
      COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
      COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
      DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCORE(*),KNPEG(*),VPREE(*),
1  KNE(*),VKE(*),VFE(*),VKGS(*),VKGD(*),VKGI(*),VFG(*),KNEQ(*),
2  KDLNC(*),VPREG(*),KSDEG(*),VDLE(*),VDLG(*),KNELS(*)
      DATA ZERO/0.0/DO/
C-----
C----- INITIALIZE VFG
      DO 10 I=1,NEQ
10  VFG(I)=ZERO
C----- READ AN ELEMENT DEWATER CARD
      KK=0
20  READ(MR,1000) IEL,IGEN,INCR
1000 FORMAT(3I5)
      IF(IEL.LE.0) GO TO 40

```

```

DO 30 IE=1,IGEN
KK=KK+1
KNELS(KK)=IEL
KSDEG(IEL)=0
30   IEL=IEL+INCR
GO TO 20
C----- ASSEMBLE NEW FG
40   CALL ASFGO(KLD,VDIMP,KLOCE,VCORE,KNPEG,VPREE,KNE,VKE,VFE,VKGS,
1     VKGD,VKG,I,VFG,VPREG,KSDEG,VDLE,VDLG,KNELS)
C----- OUTPUT
IF(M.EQ.1) WRITE(MP,2000) (VFG(I),I=1,NEQ)
2000 FORMAT('' DWAT LOAD VECTOR (FG)'/(1X,10E12.5))
WRITE(MP,2010) (KNELS(I),I=1,NELS)
2010 FORMAT('' DWAT ELEMENTS:'/(16I5))
RETURN
END
SUBROUTINE BLEXCA
C=====
C TO CALL BLOCK 'EXCA' (EXCAVATION)
C TO ASSEMBLE EQUIVALENT NODAL LOADS ON EXCAVATED SURFACE
C=====
IMPLICIT REAL*8(A-H,O-Z)
CHARACTER*4 TBL
COMMON/COOR/NDIM,NNT,NDLN,NDLT
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,LAG,NIDENT,NPG,NSEM,
1 NPEM,NELS
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
COMMON/LOCA/LCORG,LDLNC,LNEQ,LDIMP,LCODE,LINTF,LPREG,LLD,LLOCE,
1 LCORE,LNE,LPREE,LNPEG,LSDEG,LDLE,LKE,LFE,LKGS,LKGD,LKGI,LFG,
2 LRES,LDLG,LNELS,LEB,LPB
COMMON/SEQU/NSEQ,NSEQ0,ISEQ,IASSEL,IBAR
COMMON VA(20000)
DATA TBL/'NELS'/
C-----
WRITE(*,3000)
3000 FORMAT(' BLEXCA')
ISEQ=3
IASSEL=1
IBAR=0
READ(MR,1000) NELS
1000 FORMAT(I5)
WRITE(MP,2000) M,NELS
2000 FORMAT('' ASSEMBLING OF EXCAVATION LOADS (M=',I2,')''/1X,40('=')/
1 15X,'NUMBER OF EXCA ELEMENTS           (NELS)='',I5/')
IF(NELS.EQ.0) WRITE(MP,2010)
2010 FORMAT('' ** ERROR, NO ELEMENT FOR (EXCA)'')
IF(NELS.EQ.0) STOP
CALL ESPACE(NELS,0,TBL,LNELS)
CALL EXEXCA(VA(LLD),VA(LDIMP),VA(LLOCE),VA(LCORE),VA(LNPEG),
1 VA(LPREE),VA(LNE),VA(LKE),VA(LFE),VA(LKGS),VA(LKGD),VA(LKGI),
2 VA(LFG),VA(LNEQ),VA(LDLNC),VA(LPREG),VA(LSDEG),VA(LDLG),
3 VA(LDLE),VA(LRES),VA(LNELS))
CALL DSPACE(LNELS,0,TBL)

```

```

RETURN
END
SUBROUTINE EXEXCA(KLD,VDIMP,KLOCE,VCORE,KNPEG,VPREE,KNE,VKE,VFE,
1 VKGS,VKGD,VKGI,VFG,KNEQ,KDLNC,VPREG,KSDEG,VDLG,VDLE,VRES,KNELS)
C=====
C      TO EXECUTE BLOCK 'EXCA'
C      ASSEMBLE EQUIVALENT NODAL LOADS (ELEMENT FUNCTION 5)
C=====
IMPLICIT REAL*8(A-H,O-Z)
COMMON/COOR/NDIM,NNT,NDLN,NDLT
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,LAG,NIDENT,NPG,NSEM,
1 NPEM,NELS
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCORE(*),KNPEG(*),VPREE(*),
1 KNE(*),VKE(*),VFE(*),VKGS(*),VKGD(*),VKGI(*),VFG(*),KNEQ(*),
2 KDLNC(*),VPREG(*),KSDEG(*),VDLG(*),VDLE(*),VRES(*),KNELS(*)
DATA ZERO/0.D0/
C-----
C----- INITIALIZE VFG AND VRES
DO 10 I=1,NEQ
VRES(I)=ZERO
10 VFG(I)=ZERO
C----- READ 'AIR' ELEMENT PROPERTY NUMBER
READ(MR,1000) IGPE
1000 FORMAT(15)
C----- READ AN ELEMENT EXCAVATION CARD
KK=0
20 READ(MR,1100) IEL,IGEN,INCR
1100 FORMAT(3I5)
IF(IEL.LE.0) GO TO 40
DO 30 IE=1,IGEN
KK=KK+1
KNELS(KK)=IEL
30 IEL=IEL+INCR
GO TO 20
C----- ASSEMBLE NEW TOTAL FG
40 CALL ASFGT(KLD,VDIMP,KLOCE,VCORE,KNPEG,VPREE,KNE,VKE,VFE,VKGS,
1 VKGD,VKGI,VFG,VPREG,KSDEG,VDLE,VDLG,VRES,KNELS)
C----- ASSEMBLE NEW FG
CALL ASFGO(KLD,VDIMP,KLOCE,VCORE,KNPEG,VPREE,KNE,VKE,VFE,VKGS,
1 VKGD,VKGI,VFG,VPREG,KSDEG,VDLE,VDLG,KNELS)
C----- OUTPUT
IF(M.EQ.1) WRITE(MP,2000) (VFG(I),I=1,NEQ)
2000 FORMAT(' EXCAVATION LOAD VECTOR (FG)'/(1X,10E12.5))
WRITE(MP,2010) (KNELS(I),I=1,NELS)
2010 FORMAT(' EXCA ELEMENTS:'/(16I5))
C----- CHANGE EXCAVATION ELEMENT TO AIR ELEMENT
DO 60 I=1,NELS
IEL=KNELS(I)
KNPEG(IEL)=IGPE
60 CONTINUE
RETURN
END

```

```

SUBROUTINE BLEMBA
C=====
C      TO CALL BLOCK 'EMBA' (BACKFILL OR CONCRETE PLACEMENT)
C      TO ASSEMBLE EQUIVALENT NODAL LOADS DUE TO EMBANKMENT
C=====

IMPLICIT REAL*8(A-H,O-Z)
CHARACTER*4 TBL
COMMON/COOR/NDIM,NNT,NDLN,NDLT
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1 NPEM,NELS
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
COMMON/LOCA/LCORG,LDLNC,LNEQ,LDIMP,LCODE,LINTF,LPREG,LLD,LLOC,
1 LCORE,LNE,LPREE,LNPEG,LSDEG,LDLE,LKE,LFE,LKGS,LKGD,LKGI,LFG,
2 LRES,LDLG,LNELS,LEB,LPB
COMMON/SEQU/NSEQ,NSEQO,ISEQ,IASSEL,IBAR
COMMON VA(20000)
DATA TBL/'NELS'/

C-----
3000 WRITE(*,3000)
FORMAT(' BLEMBA')
ISEQ=4
IASSEL=1
IBAR=0
READ(MR,1000) NELS
1000 FORMAT(I5)
WRITE(MP,2000) M,NELS
2000 FORMAT(//' ASSEMBLING OF EMBANKMENT LOADS (M=',I2,')/1X,40('=')/
1 15X,'NUMBER OF EMBA ELEMENTS' (NELS)=',I5/)
IF(NELS.EQ.0) WRITE(MP,2010)
2010 FORMAT(//'* ERROR, NO ELEMENT FOR EMBA')
IF(NELS.EQ.0) STOP
CALL ESPACE(NELS,0,TBL,LNELS)
CALL EXEMBA(VA(LLD),VA(LDIMP),VA(LLOC),VA(LCORE),VA(LNPEG),
1 VA(LPREE),VA(LNE),VA(LKE),VA(LFE),VA(LKGS),VA(LKGD),VA(LKGI),
2 VA(LFG),VA(LNEQ),VA(LLDNC),VA(LPREG),VA(LSDEG),VA(LDLE),
3 VA(LDLG),VA(LNELS))
CALL DSPACE(LNELS,0,TBL)
RETURN
END

SUBROUTINE EXEMBA(KLD,VDIMP,KLOCE,VCORE,KNPEG,VPREE,KNE,VKE,VFE,
1 VKGS,VKGD,VKGI,VFG,KNEQ,KDLNC,VPREG,KSDEG,VDLE,VDLG,KNELS)
C=====
C      TO EXECUTE BLOCK 'EMBA'
C      ASSEMBLE EQUIVALENT NODAL LOADS (ELEMENT FUNCTION 4)
C=====

IMPLICIT REAL*8(A-H,O-Z)
COMMON/COOR/NDIM,NNT,NDLN,NDLT
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1 NPEM,NELS
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCORE(*),KNPEG(*),VPREE(*),
1 KNE(*),VKE(*),VFE(*),VKGS(*),VKGD(*),VKGI(*),VFG(*),KNEQ(*),
2 KDLNC(*),VPREG(*),KSDEG(*),VDLE(*),VDLG(*),KNELS(*)

```

```

      DATA ZERO/O.O/DO/
C----- INITIALIZE VFG
C----- DO 10 I=1,NEQ
10    VFG(I)=ZERO
C----- READ AN ELEMENT EMBANKMENT CARD
      KK=0
20    READ(MR,1000) IEL,IGEN,INCR,IGPE
1000  FORMAT(4I5)
      IFIEL.LE.0) GO TO 40
      DO 30 IE=1,IGEN
      KK=KK+1
      KNELS(KK)=IEL
      KNPEG(IEL)=IGPE
30    IEL=IEL+INCR
      GO TO 20
C----- ASSEMBLE NEW FG
40    CALL ASFGO(KLD,VDIMP,KLOCE,VCORE,KNPEG,VPREE,KNE,VKE,VFE,VKGS,
1      VKGD,VKG,I,VFG,VPREG,KSDEG,VDLE,VDLG,KNELS)
C----- OUTPUT
      IF(M.EQ.1) WRITE(MP,2000) (VFG(I),I=1,NEQ)
2000  FORMAT(' EMBA LOAD VECTOR (FG)''/1X,10E12.5))
      WRITE(MP,2010) (KNELS(I),I=1,NELS)
2010  FORMAT(' EMBA ELEMENTS:'/(16I5))
      RETURN
      END
      SUBROUTINE BLFWAT
C===== TO CALL BLOCK 'FWAT' (RAISE WATER TABLE LEVEL)
C===== TO ASSEMBLE EQUIVALENT NODAL LOADS DUE TO RAISE OF WATER TABLE
C=====
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*4 TBL
      COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1      NPEM,NELS
      COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
      COMMON/LOCA/LCORG,LDLNC,LNEQ,LDIMP,LCODE,LINTF,LPREG,LLD,LLOCE,
1      LCORE,LNE,LPREE,LNPEG,LSDEG,LDLE,LKE,LFE,LKGS,LKGD,LKGI,LFG,
2      LRES,LDLG,LNELS,LEB,LPB
      COMMON/SEQU/NSEQ,NSEQO,ISEQ,IASSEL,IBAR
      COMMON VA(20000)
      DATA TBL/'NELS'/
C----- WRITE(*,3000)
3000  FORMAT(' BLFWAT')
      ISEQ=8
      IASSEL=0
      IBAR=0
      READ(MR,1000) NELS
1000  FORMAT(I5)
      WRITE(MP,2000) M,NELS
2000  FORMAT(' ASSEMBLING OF FWAT LOADS (M=',I2,')'/1X,40('=')/
1      15X,'NUMBER OF FWAT ELEMENTS'          (NELS)=' ,I5/)
      IF(NELS.EQ.0) WRITE(MP,2010)

```

```

2010 FORMAT(//' ** ERROR, NO ELEMENT FOR FWAT!/')
  IF(NELS.EQ.0) STOP
  CALL ESPACE(NELS,0,TBL,LNELS)
  CALL EXFWAT(VA(LLD),VA(LDIMP),VA(LLOCE),VA(LCORE),VA(LNPEG),
1   VA(LPREE),VA(LNE),VA(LKE),VA(LFE),VA(LKGS),VA(LKGD),VA(LKGI),
2   VA(LFG),VA(LNEQ),VA(LDLNC),VA(LPREG),VA(LSDEG),VA(LDLE),
3   VA(LDLG),VA(LNELS))
  CALL DSPACE(LNELS,0,TBL)
  RETURN
END

SUBROUTINE EXFWAT(KLD,VDIMP,KLOCE,VCORE,KNPEG,VPREE,KNE,VKE,VFE,
1   VKGS,VKGD,VKGI,VFG,KNEQ,KDLNC,VPREG,KSDEG,VDLE,VDLG,KNELS)
C=====
C      TO EXECUTE BLOCK 'FWAT'
C      ASSEMBLE EQUIVALENT NODAL LOADS (ELEMENT FUNCTION 6)
C=====

IMPLICIT REAL*8(A-H,O-Z)
COMMON/COOR/NDIM,NNT,NDLN,NDLT
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1  NPEN,NELS
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDEL
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCORE(*),KNPEG(*),VPREE(*),
1  KNE(*),VKE(*),VFE(*),VKGS(*),VKGD(*),VKGI(*),VFG(*),KNEQ(*),
2  KDLNC(*),VPREG(*),KSDEG(*),VDLE(*),VDLG(*),KNELS(*)
DATA ZERO/0.D0/
C-----
C      INITIALIZE VFG
DO 10 I=1,NEQ
10  VFG(I)=ZERO
C----- READ AN ELEMENT FWAT CARD
KK=0
20  READ(MR,1000) IEL,IGEN,INCR
1000 FORMAT(3I5)
IFIEL.LE.0) GO TO 40
DO 30 IE=1,IGEN
KK=KK+1
KNELS(KK)=IEL
KSDEG(IEL)=1
30  IEL=IEL+INCR
GO TO 20
C----- ASSEMBLE NEW FG
40  CALL ASFGO(KLD,VDIMP,KLOCE,VCORE,KNPEG,VPREE,KNE,VKE,VFE,VKGS,
1   VKGD,VKGI,VFG,VPREG,KSDEG,VDLE,VDLG,KNELS)
C----- OUTPUT
IF(M.EQ.1) WRITE(MP,2000) (VFG(I),I=1,NEQ)
2000 FORMAT(/' FWAT LOAD VECTOR (FG)'/(1X,10E12.5))
WRITE(MP,2010) (KNELS(I),I=1,NELS)
2010 FORMAT(/' FWAT ELEMENTS:'/(16I5))
RETURN
END

SUBROUTINE ASFGT(KLD,VDIMP,KLOCE,VCORE,KNPEG,VPREE,KNE,VKE,VFE,
1   VKGS,VKGD,VKGI,VFG,VPREG,KSDEG,VDLE,VDLG,VRES,KNELS)
C=====

```

```

C      UPDATE AND ASSEMBLE TOTAL EQUIVALENT NODAL LOADS AFTER DELETING
C      THE EXCAVATED FREE BODY
C=====
C      IMPLICIT REAL*8(A-H,O-Z)
COMMON/COOR/NDIM,NNT,NDLN,NDLT
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1 NPEM,NELS
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDL
COMMON/RGDTIEL,IKEL,ITPE,ITPE1,ISDE,IMODEL,IFEL,IDL,ICE,INEL,
1 IPG,ICODE,IMATD,INSE,INPE,IDELO,INELO,IPGO,IRES
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCORE(*),KNPEG(*),VPREE(*),
1 KNE(*),VKE(*),VFE(*),VKGS(*),VKGD(*),VKGI(*),VFG(*),VPREG(*),
2 KSDEG(*),VDLE(*),VDLG(*),VRES(*),KNELS(*)
C=====
C----- LOOP OVER SPECIFIED ELEMENTS
DO 30 N=1,NELS
C----- READ AN ELEMENT FROM FILE ME
IEL=KNELS(N)
CALL RDELEM(ME,KLOCE,VCORE,KNE)
C----- EVALUATE INTERPOLATION FUNCTIONS IF REQUIRED
IF(ITPE.EQ.ITPE1) GO TO 20
ICODE=2
CALL ELEMELB(VCORE,VPREE,VDLE,VKE,VFE)
C----- EVALUATE ELEMENT NODAL FORCES DUE TO STRESSES AT G.P.
20 IF(NLAG.GT.0) CALL DLELM(KLOCE,VDLG,VDIMP,VDLE)
ICODE=9
CALL ELEMELB(VCORE,VPREE,VDLE,VKE,VFE)
C----- ASSEMBLE
CALL ASSEL(0,1,IDL,NSYM,KLOCE,KLD,VKE,VFE,VKGS,VKGD,VKGI,VRES)
30 ITPE1=ITPE
C----- UPDATE TOTAL LOAD VECTOR
DO 40 I=1,NEQ
40 VRES(NDLT+I)=VRES(NDLT+I)-VRES(I)
RETURN
END
SUBROUTINE ASFGO(KLD,VDIMP,KLOCE,VCORE,KNPEG,VPREE,KNE,VKE,VFE,
1 VKGS,VKGD,VKGI,VFG,VPREG,KSDEG,VDLE,VDLG,KNELS)
C=====
C ASSEMBLE EQUIVALENT NODAL LOADS FOR EXCAVATION AND EMBANKMENT
C=====
C      IMPLICIT REAL*8(A-H,O-Z)
COMMON/PREL/NGPE,NPRE,NPRM,NMNL,NHIS
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1 NPEM,NELS
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDL
COMMON/RGDTIEL,IKEL,ITPE,ITPE1,ISDE,IMODEL,IFEL,IDL,ICE,INEL,
1 IPG,ICODE,IMATD,INSE,INPE,IDELO,INELO,IPGO,IRES
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
COMMON/SEQU/NSEQ,NSEQ0,ISEQ,IASSEL,IBAR
DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCORE(*),KNPEG(*),VPREE(*),
1 KNE(*),VKE(*),VFE(*),VKGS(*),VKGD(*),VKGI(*),VFG(*),VPREG(*),
2 KSDEG(*),VDLE(*),VDLG(*),KNELS(*)
C

```

```

C----- LOOP OVER THE SPECIFIED ELEMENTS
DO 30 N=1,NELS
C----- READ AN ELEMENT FROM FILE ME
IEL=KNELS(N)
WRITE(*,3000) IEL
3000 FORMAT(' ELEMENT LOAD VECTOR',I5)
CALL RDELEM(ME,KLOCE,VCORE,KNE)
IC=(KNPEG(IEL)-1)*(NPRE+NPRM)
DO 10 I=1,NPRE+NPRM
10  VPREE(I)=VPREG(IC+I)
C----- EVALUATE INTERPOLATION FUNCTIONS IF REQUIRED
IF(ITPE.EQ.ITPE1) GO TO 20
ICODE=2
CALL ELEMLB(VCORE,VPREE,VDLE,VKE,VFE)
C----- GENERATE ELEMENT D.O.F. FOR LAGRANGIAN FORMULATION
20  IF(NLAG.GT.0) CALL DLELM(KLOCE,VDLG,VDIMP,VDLE)
C----- EVALUATE ELEMENT VECTOR
ICODE=4
IF(IEQ.EQ.3) ICODE=5
CALL ELEMLB(VCORE,VPREE,VDLE,VKE,VFE)
C----- PRINT ELEMENT VECTOR VFE
IF(M.GE.2) WRITE(MP,2000) IEL,(VFE(I),I=1,1DLE)
2000 FORMAT('// VECTOR (FE) , ELEMENT:',I5/(10X,10E12.5))
C----- ASSEMBLE
CALL ASSEL(0,1,1DLE,NSYM,KLOCE,KLD,VKE,VFE,VKGS,VKGD,VKGI,VFG)
30  ITPE1=ITPE
RETURN
END
SUBROUTINE BLBARA
C=====
C   TO CALL BLOCK 'BARA'
C   TO MODIFY GLOBAL ELEMENT PROPERTY TABLE (KNPEG) AFTER ADDING BARS
C=====
IMPLICIT REAL*8(A-H,O-Z)
CHARACTER*4 TBL
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1 NPEM,NELS
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
COMMON/LOCA/LCORG,LDLNC,LNEQ,LDIMP,LCODE,LINTF,LPREG,LLD,LLOCE,
1 LCORE,LNE,LPREE,LNPEG,LSDEG,LDE,LKE,LFE,LKGS,LKGD,LKGI,LFG,
2 LRES,LDLG,LNELS,LEB,LPB
COMMON/SEQU/NSEQ,NSEQ0,IEQ,IASSEL,IBAR
COMMON VA(20000)
DATA TBL/'NELS'/
C-----
IEQ=5
IBAR=1
READ(MR,1000) NELS
1000 FORMAT(I5)
WRITE(MP,2000) M,NELS
2000 FORMAT('// ADDING BAR ELEMENTS (M=',I2,')'/1X,40('='))
1 15X,'NUMBER OF ADDED BAR ELEMENTS          (NELS)=',I5/
IF(NELS.EQ.0) WRITE(MP,2010)
2010 FORMAT('// ** ERROR, NO BAR IS ADDED (NBAR=0)'/>

```

```

IF(NELS.EQ.0) STOP
CALL ESPACE(NELS,0,TBL,LNELS)
CALL EXBARA(VA(LNPEG),VA(LNELS))
CALL DSPACE(LNELS,0,TBL)
RETURN
END
SUBROUTINE EXBARA(KNPEG,KNELS)
C=====
C TO EXECUTE BLOCK 'BARA' AND MODIFY MATRIX KNPEG
C=====
IMPLICIT REAL*8(A-H,O-Z)
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1 NPEM,NELS
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
DIMENSION KNPEG(*),KNELS(*)
C-----
I=1
10 READ(MR,1000) IEL,IGPE
1000 FORMAT(2I5)
IF(IEL.LE.0) GO TO 20
KNELS(I)=IEL
KNPEG(IEL)=IGPE
I=I+1
GO TO 10
20 WRITE(MP,2000) (KNELS(I),I=1,NELS)
2000 FORMAT(' ADDED BAR ELEMENTS:'/(1X,16I5))
RETURN
END
SUBROUTINE BLBARD
C=====
C TO CALL BLOCK 'BARD'
C GENERATE CONCENTRATED LOADS DUE TO DELETION OF BAR ELEMENTS
C=====
IMPLICIT REAL*8(A-H,O-Z)
CHARACTER*4 TBL
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1 NPEM,NELS
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
COMMON/LOCA/LCORG,LDLNC,LNEQ,LDIMP,LCODE,LINTF,LPREG,LLD,LLOCE,
1 LCORE,LNE,LPREE,LNPEG,LDLNG,LSDEG,LDLE,LKE,LFE,LKGS,LKGD,LKGI,LFG,
2 LRES,LDLG,LNELS,LEB,LPB
COMMON/SEQU/NSEQ,NSEQO,ISEQ,IASSEL,IBAR
COMMON VA(20000)
DATA TBL/'NELS'/
C-----
ISEQ=6
IASSEL=1
IBAR=0
READ(MR,1000) NELS
1000 FORMAT(I5)
WRITE(MP,2000) M,NELS
2000 FORMAT(' ASSEMBLING OF BAR-DELETION LOADS (M=',I2,')'/
1 1X,40('=')/15X,'NUMBER OF DELETED BAR ELEMENTS (NELS)=',I5/)
IF(NELS.EQ.0) WRITE(MP,2010)

```

```

2010 FORMAT(//' ERROR, NO BAR IS DELETED (NELS=0)')/
  IF(NELS.EQ.0) STOP
  CALL ESPACE(NELS,0,TBL,LNELS)
  CALL EXBARD(VA(LFG),VA(LCORE),VA(LPREG),VA(LPREE),VA(LDLNC),
  1 VA(LLOCE),VA(LNE),VA(LKE),VA(LFE),VA(LDLE),VA(LKGS),VA(LKGD),
  2 VA(LKG),VA(LLD),VA(LNEQ),VA(LNPEG),VA(LNELS))
  CALL DSPACE(LNELS,0,TBL)
  RETURN
  END
  SUBROUTINE EXBARD(VFG,VCORE,VPREG,VPREE,KDLNC,KLOCE,KNE,VKE,VFE,
  1 VDLE,VKGS,VKGD,VKGI,KLD,KNEQ,KNPEG,KNELS)
C=====
C      TO EXECUTE BLOCK 'BARD'
C      GENERATE CONCENTRATED LOADS AND UPDATE VECTOR KNPEG
C=====
IMPLICIT REAL*8(A-H,O-Z)
COMMON/PREL/NGPE,NPRE,NPRM,NMNL,NHIS
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
 1 NPEM,NELS
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
COMMON/RGDTIEL,IKEL,ITPE,ITPE1,ISDE,IMODEL,IFEL,IDLE,ICE,INEL,
 1 IPC,ICODE,IMATD,INSE,INPE,IDELO,INELO,IPGO,IRES
 1 KNE(*),VKE(*),VFE(*),VDLE(*),VKGS(*),VKGD(*),VKGI(*),KLD(*),
 2 KNEQ(*),KNPEG(*),KNELS(*)
  DATA ZERO/0.0/DO/
C-----
C----- INITIALIZE VFG
  DO 10 I=1,NEQ
  10 VFG(I)=ZERO
  IN=1
  20 READ(MR,1000) IEL
  1000 FORMAT(I5)
  IF(IEL.EQ.0) GO TO 40
  KNELS(IN)=IEL
C----- READ ELEMENT IEL FROM FILE ME
  CALL RDELEM(ME,KLOCE,VCORE,KNE)
  IC=(KNPEGIEL-1)*(NPRE+NPRM)
  DO 30 I=1,NPRE+NPRM
  30 VPREE(I)=VPREG(IC+I)
  ICODE=5
  CALL ELEMELB(VCORE,VPREE,VDLE,VKE,VFE)
  CALL ASSEL(0,1,IDL,NSYM,KLOCE,KLD,VKE,VFE,VKGS,VKGD,VKGI,VFG)
C----- UPDATE NEW KNPEG
  KNPEGIEL=0
  IN=IN+1
  GO TO 20
  40 WRITE(MP,2000) (KNELS(I),I=1,NELS)
  2000 FORMAT(' DELETED BAR ELEMENTS:'/(16I5))
  IF(M.GE.1) WRITE(MP,2010) (VFG(I),I=1,NEQ)
  2010 FORMAT(' BARD LOAD VECTOR (FG)'/(1X,10E12.5))
  RETURN
  END

```

```

SUBROUTINE BLLINM
C=====
C   TO CALL BLOCK 'LINM'
C   ASSEMBLE AND SOLVE A LINEAR PROBLEM IN CORE
C=====
IMPLICIT REAL*8(A-H,O-Z)
CHARACTER*4 TBL
COMMON/COOR/NDIM,NNT,NDLN,NDLT
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1 NPEM,NELS
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
COMMON/NUMB/M,1,ME,M3,M4,MR,MP,M7,MS,M9,MH
COMMON/LOCA/LCORG,LDLNC,LNEQ,LDIMP,LCODE,LINTF,LPREG,LLD,LLOCE,
1 LCORE,LNE,LPREE,LNPEG,LSDEG,LDLE,LKE,LFE,LKGS,LKGD,LKGI,LFG,
2 LRES,LDLG,LNELS,LEB,LPB
COMMON/SEQ/NESEQ,NSEQO,ISEQ,IASSEL,IBAR
COMMON/NLIN/EPSDL,OMEGA,DPAS,NPAS,NITER,ITER,IMETH,ILAG,INLN
COMMON VA(20000)
DIMENSION TBL(9)
DATA TBL/'KGS ','KGD ','KGI ','FG ','KE ','FE ','RES ','DLE ',
1 'DLG '/
C-----
      ILAG=0
      INLN=0
      NSEQ=NSEQ+1
      WRITE(MP,1000) M
1000 FORMAT('' ASSEMBLING AND LINEAR SOLUTION (M=',I2,')'' ',30('='))
      IF(NSEQ.NE.1) GO TO 10
C----- OPEN DIRECT ACCESS FILE MS (STRESSES, STRAINS & PWPS)
      LS=NSEM*8
      OPEN(MS,FILE='MS.DAT',STATUS='NEW',ACCESS='DIRECT',RECL=LS)
C----- TO ALLOCATE SPACE
      IF(LKGS.EQ.1) CALL ESPACE(NKG,1,TBL(1),LKGS)
      IF(LKGD.EQ.1) CALL ESPACE(NEQ,1,TBL(2),LKGD)
      IF(NSYM.EQ.1.AND.LKGI.EQ.1) CALL ESPACE(NKG,1,TBL(3),LKGI)
      IF(LFG.EQ.1) CALL ESPACE(NEQ,1,TBL(4),LFG)
      IF(LKE.EQ.1) CALL ESPACE(NKE,1,TBL(5),LKE)
      IF(LFE.EQ.1) CALL ESPACE(NDLE,1,TBL(6),LFE)
      IF(LRES.EQ.1) CALL ESPACE(NDLT-NEQ,1,TBL(7),LRES)
      IF(LDLE.EQ.1) CALL ESPACE(NDLE,1,TBL(8),LDLE)
      IF(LDLG.EQ.1) CALL ESPACE(NEQ,1,TBL(9),LDLG)
C----- TO EXECUTE THE BLOCK
10     CALL EXLINMC(VA(LLD),VA(LDIMP),VA(LLOCE),VA(LCORE),VA(LCODE),
1     VA(LINTF),VA(LPREG),VA(LPREE),VA(LNE),VA(LKE),VA(LFE),VA(LKGS),
2     VA(LKGD),VA(LKGI),VA(LFG),VA(LCORG),VA(LDLNC),VA(LNEQ),
3     VA(LNPEG),VA(LRES),VA(LSDEG),VA(LDLE),VA(LDLG))
      RETURN
      END
      SUBROUTINE EXLINM(KLD,VDIMP,KLOCE,VCORE,KCODE,KINTF,VPREG,VPREE,
1     KNE,VKE,VFE,VKGS,VKGD,VKGI,VFG,VCORG,KDLNC,KNEQ,KNPEG,VRES,
2     KSDEG,VDLE,VDLG)
C=====
C   TO EXECUTE BLOCK 'LINM'
C   ASSEMBLE AND SOLVE A LINEAR PROBLEM IN CORE

```

```

C=====
IMPLICIT REAL*(A-H,O-Z)
COMMON/COOR/NDIM,NNT,NDLN,NDLT
COMMON/COND/NCLT,NCLZ,NCLNZ
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1 NPEM,NELS
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
COMMON/NUMB/M,1,ME,M3,M4,MR,MP,M7,MS,M9,MH
COMMON/SEQN/NSEQ,NSEQ0,ISEQ,IASSEL,IBAR
DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCORE(*),KCODE(*),KINTF(*),
1 VPREG(*),VPREE(*),KNE(*),VKE(*),VFE(*),VKGS(*),VKGD(*),VKGI(*),
2 VFG(*),VCORG(*),KDLNC(*),KNEQ(*),KNPEG(*),VRES(*),KSDEG(*),
3 VDLE(*),VDLG(*)
DATA ZERO/0.0/D/
C-----
C----- INITIALIZE FILE MS
IF(NSEQ.EQ.1) CALL INITMS(NELT,NSEM,MS)
C----- SAVE UNMODIFIED VECTOR VFG (BY B.C.) IN VECTOR VRES
DO 10 I=1,NEQ
10 VRES(NDLT+I)=VRES(NDLT+I)+VFG(I)
IF(M.GE.2) WRITE(MP,2000) (VFG(I),I=1,NEQ)
2000 FORMAT(' GLOBAL LOAD VECTOR NON MODIFIED BY B.C. (FG)' 
1/(1X,10E12.5))
C----- DETERMINE IF THE GLOBAL MATRIX IS TO BE ASSEMBLED
IKT=0
IF(NCLNZ.NE.0) IKT=1
IF(IASSEL.EQ.1) IKT=1
IF(IKT.EQ.0) GO TO 50
C----- DETERMINE IF THE GLOBAL MATRIX IS TO BE INITIALIZED
IF(NSEQ.EQ.1) GO TO 40
DO 20 I=1,NKG
VKGS(I)=ZERO
20 IF(NSYM.EQ.1) VKGI(I)=ZERO
DO 30 I=1,NEQ
30 VKGD(I)=ZERO
C----- ASSEMBLE KG, MODIFY FG FOR THE B.C.
40 CALL ASKG(KLD,VDIMP,KLOCE,VCORE,KCODE,KINTF,VPREG,VPREE,KNE,
1 VKE,VFE,VKGS,VKGD,VKGI,VFG,KNPEG,KSDEG,VDLE,VDLG)
IF(ISEQ.NE.0) GO TO 50
C----- PRINT KG AND FG
IF(M.LT.2) GO TO 50
WRITE(MP,2005) (VKGS(I),I=1,NKG)
2005 FORMAT(' GLOBAL MATRIX (KG)''/      UPPER TRIANGLE'/
1 (1X,10E12.5))
WRITE(MP,2010) (VKGD(I),I=1,NEQ)
2010 FORMAT('      DIAGONAL'/(1X,10E12.5))
IF(NSYM.EQ.1) WRITE(MP,2020) (VKGI(I),I=1,NKG)
2020 FORMAT('      LOWER TRIANGLE'/(1X,10E12.5))
WRITE(MP,2030) (VFG(I),I=1,NEQ)
2030 FORMAT(' GLOBAL LOAD VECTOR MODIFIED BY THE B.C. (FG)' 
1 /(1X,10E12.5))
C----- SOLVE
50 CALL SOL(VKGS,VKGD,VKGI,VFG,KLD,NEQ,MP,IKT,1,NSYM,ENERG)
IF(ISEQ.NE.0) GO TO 60

```

```

      IF(NSYM.NE.1) WRITE(MP,2040) ENERG
2040 FORMAT(15X,'ENERGY      (ENERG)=',1E12.5)
      IF(M.LT.2) GO TO 60
      WRITE(MP,2050) (VKGS(I),I=1,NKG)
2050 FORMAT('' TRIANGULARIZED MATRIX (KG)'/'    UPPER TRIANGLE'/
1   (1X,10E12.5))
      WRITE(MP,2010) (VKGD(I),I=1,NEQ)
      IF(NSYM.EQ.1) WRITE(MP,2020) (VKGI(I),I=1,NKG)
C----- PIVOTS OF KG AND DETERMINANT
      CALL PRPVTS(VKGD)
C----- UPDATE THE SOLUTION
60      DO 70 I=1,NEQ
70      VDLG(I)=VDLG(I)+VFG(I)
C----- PRINT THE SOLUTION
      WRITE(MP,2060)
2060 FORMAT('' SOLUTION'')
      CALL PRSOL(KDLNC,VCORG,VDIMP,KNEQ,VDLG)
C----- EVALUATE AND UPDATE STRESSES, STRAINS AND PWPS
      CALL ASGRAD(KLD,VDIMP,KLOCE,VCORE,KCODE,KINTF,VPREG,VPREE,KNE,
1   VKE,VFE,VKGS,VKGD,VKGI,VFG,KNPEG,KSDEG,VDLE,VDLG,NRES)
      IF(ISEQ.NE.1) GO TO 90
      DO 80 I=1,NEQ
80      VDLG(I)=ZERO
C----- FOR THE PROBLEM WITHOUT CONSTRUCTION SEQUENCES
C----- EVALUATE AND PRINT EQUILIBRIUM RESIDUAL VECTOR
C----- ASSEMBLE THE RESIDUALS
90      IF(ISEQ.NE.0) RETURN
      DO 100 I=1,NEQ
100     VRES(I)=-VRES(NDLT+I)
      CALL ASRESD(1,1,KLD,VDIMP,KLOCE,VCORE,VPREG,VPREE,KNE,VKE,VFE,
1   VKGS,VKGD,VKGI,VFG,KNPEG,KSDEG,VDLE,VDLG,VRES,VRES(NEQ+1))
C----- PRINT THE RESIDUALS
      WRITE(MP,2070)
2070 FORMAT('' EQUILIBRIUM RESIDUALS AND REACTIONS'')
      CALL PRSOL(KDLNC,VCORG,VRES(NEQ+1),KNEQ,VRES)
      RETURN
      END
      SUBROUTINE ASKG(KLD,VDIMP,KLOCE,VCORE,KCODE,KINTF,VPREG,VPREE,
1   KNE,VKE,VFE,VKGS,VKGD,VKGI,VFG,KNPEG,KSDEG,VDLE,VDLG)
C=====C=====C=====C=====C=====C=====C=====C=====
C   TO ASSEMBLE GLOBAL MATRIX KG (ELEMENT FUNCTION3)
C   TAKING INTO ACCOUNT OF NON ZERO PRESCRIBED D.O.F.
C=====C=====C=====C=====C=====C=====C=====C=====
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/COND/NCLT,NCLZ,NCLNZ
      COMMON/PREL/NGPE,NPRE,NPRM,NMNL,NHIS
      COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1   NPEM,NELS
      COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
      COMMON/RGDT/IEL,IKEL,ITPE,ITPE1,ISDE,IMODEL,IFEL,IDELE,ICE,INEL,
1   IPG,ICODE,IMATD,INSE,INPE,IDELE,INELO,IPGO,IRES
      COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
      COMMON/NLIN/EPSDL,OMEGA,DPAS,MPAS,NITER,ITER,IMETH,ILAG,INLN
      DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCORE(*),KCODE(*),KINTF(*),

```

```

1  VPREG(*),VPREE(*),KNE(*),VKE(*),VFE(*),VKGS(*),VKGD(*),
2  VKGI(*),VFG(*),KNPEG(*),KSDEG(*),VDLE(*),VDLG(*)
C-----
C----- LOOP OVER THE ELEMENTS
DO 30 IEL=1,NELT
IGPE=KNPEGIEL)
WRITE(*,3000) IEL
3000 FORMAT(' ELEMENT STIFFNESS MATRIX',I5)
C----- READ AN ELEMENT ON FILE M2
CALL RDELEM(ME,KLOCE,VCORE,KNE)
IC=(IGPE-1)*(NPRE+NPRM)
DO 10 I=1,NPRE+NPRM
10  VPREE(I)=VPREG(IC+I)
IMODEL=KCODE(IGPE)
IFEL=KINTF(IGPE)
IF(NSDG.GT.0) ISDE=KSDEG(IFEL)
C----- SKIP COMPUTATION IF IDENTICAL ELEMENTS ENCOUNTERED
IF(NIDENT.EQ.1.AND.IEL.GT.1) GO TO 20
C----- EVALUATE INTERPOLATION FUNCTIONS IF REQUIRED
IF(ITPE.EQ.ITPE1) GO TO 15
ICODE=2
CALL ELEMLB(VCORE,VPREE,VDLE,VKE,VFE)
C----- GENERATE ELEMENT D.O.F. FOR LAGRANGIAN FORMULATION
15  IF(ILAG.GT.0) CALL DLELM(KLOCE,VDLG,VDIMP,VDLE)
C----- FORM ELEMENT MATRIX
ICODE=3
CALL ELEMLB(VCORE,VPREE,VDLE,VKE,VFE)
C----- PRINT ELEMENT MATRIX
IF(M.LT.2) GO TO 20
IF(NSYM.EQ.0) IKE=IDLE*(IDLE+1)/2
IF(NSYM.EQ.1) IKE=IDLE*IDLE
WRITE(MP,2000) IEL,(VKE(I),I=1,IKE)
2000 FORMAT(/ MATRIX (KE) , ELEMENT:,I5/(10X,10E12.5))
C----- MODIFY FG FOR NON ZERO PRESCRIBED D.O.F. (LINEAR)
20  IF(NCLNZ.NE.0) CALL MODFG(IDLE,NSYM,KLOCE,VDIMP,VKE,VFG)
C----- ASSEMBLE
CALL ASSEL(1,0,IDLE,NSYM,KLOCE,KLD,VKE,VFE,VKGS,VKGD,VKGI,VFG)
30  ITPE1=ITPE
RETURN
END
SUBROUTINE ASGRAD(KLD,VDIMP,KLOCE,VCORE,KCODE,KINTF,VPREG,VPREE,
1 KNE,VKE,VFE,VKGS,VKGD,VKGI,VFG,KNPEG,KSDEG,VDLE,VDLG,NRES)
C=====
C----- TO EVALUATE AND PRINT GRADIENTS (STRESSES AT ELEMENT G.P.
C----- (ELEMENT FUNCTION 9)
C=====
IMPLICIT REAL*8(A-H,O-Z)
COMMON/PREL/NGPE,NPRE,NPRM,NMNL,NHIS
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1 NPEM,NELS
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
COMMON/RGDT/IEL,IKEL,ITPE,ITPE1,ISDE,IMODEL,IFEL,IDL,ICE,INEL,
1 IPG,ICODE,IMATD,INSE,INPE,IDELO,INELO,IPGO,IRES
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH

```

```

COMMON/NLIN/EPSDL,OMEGA,DPAS,NPAS,NITER,ITER,IMETH,ILAG,INLN
DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCORE(*),KCODE(*),KINTF(*),
1 VPREG(*),VPREE(*),KNE(*),VKE(*),VFE(*),VKGS(*),VKGD(*),
2 VKGI(*),VFG(*),KNPEG(*),KSDEG(*),VDLE(*),VDLG(*)
C-----
      NRES=0
C----- LOOP OVER THE ELEMENTS
      DO 50 IEL=1,NELT
      IGPE=KNPEG(IEL)
C----- READ THE ELEMENT
      CALL RDELEM(ME,KLOCE,VCORE,KNE)
      IC=(IGPE-1)*(NPRE+NPRM)
      DO 10 I=1,NPRE+NPRM
      10 VPREE(I)=VPREG(IC+I)
      IMODEL=KCODE(IGPE)
      IFEL=KINTF(IGPE)
      IF(NSDG.GT.0) ISDE=KSDEG(IEL)
C----- EVALUATE INTERPOLATION FUNCTION IF REQUIRED
      IF(ITPE.EQ.ITPE1) GO TO 30
      ICODE=2
      CALL ELEMELB(VCORE,VPREE,VDLE,VKE,VFE)
C----- GENERATE ELEMENT (TOTAL) D.O.F. FOR LAGRANGIAN FORMULATION
C----- NOTE VFE IS USED TO STORE ELEMENT NODAL DISPLACEMENTS
      30 IF(ILAG.GT.0) CALL DLELM(KLOCE,VDLG,VDIMP,VFE)
C----- FIND ELEMENT (INCREMENTAL) D.O.F.
      CALL DLELM(KLOCE,VFG,VDIMP,VDLE)
C----- COMPUTE STRESSES, STRAINS AND PWPS
      ICODE=8
      CALL ELEMELB(VCORE,VPREE,VDLE,VKE,VFE)
      IF(IRES.EQ.1) NRES=1
      50 ITPE1=ITPE
      IF(ILAG.EQ.1) NRES=1
      RETURN
      END
      SUBROUTINE ASRESD(IRESD,IREAC,KLD,VDIMP,KLOCE,VCORE,VPREG,VPREE,
1 KNE,VKE,VFE,VKGS,VKGD,VKGI,VFG,KNPEG,KSDEG,VDLE,VDLG,VRES,VREAC)
C=====
C TO ASSEMBLE INTERNAL RESIDUALS IN VRES (IF IRESD .EQ. 1)
C AND EXTERNAL REACTIONS IN VREAC (IF IREAC .EQ. 1)
C=====
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/PREL/NGPE,NPRE,NPRM,NMNL,NHIS
      COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1 NPEM,NELS
      COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
      COMMON/RGDTIEL,IKEL,ITPE,ITPE1,ISDE,IMODEL,IFEL,IDL,E,ICE,INEL,
1 IPG,ICODE,IMATD,INSE,INPE,IDELO,INEL0,IPGO,IRES
      COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
      COMMON/NLIN/EPSDL,OMEGA,DPAS,NPAS,NITER,ITER,IMETH,ILAG,INLN
      DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCORE(*),VPREG(*),VPREE(*),
1 KNE(*),VKE(*),VFE(*),VKGS(*),VKGD(*),VKGI(*),VFG(*),KNPEG(*),
2 KSDEG(*),VDLE(*),VDLG(*),VRES(*),VREAC(*)
C-----
C----- LOOP OVER THE ELEMENTS

```

```

DO 60 IEL=1,NELT
C----- READ AN ELEMENT ON FILE ME
    CALL RDELEM(ME,KLOCE,VCORE,KNE)
    IF(NSDG.GT.0) ISDE=KSDEG(IEL)
C----- EVALUATE INTERPOLATION FUNCTION IF REQUIRED
    IF(ITPE.EQ.ITPE1) GO TO 10
    ICODE=2
    CALL ELEMLB(VCORE,VPREE,VDLE,VKE,VFE)
C----- GENERATE ELEMENT D.O.F. FOR LAGRANGIAN FORMULATION
10   IF(ILAG.GT.0) CALL DLELM(KLOCE,VDLG,VDIMP,VDLE)
C----- EVALUATE ELEMENT REACTIONS
    ICODE=9
    CALL ELEMLB(VCORE,VPREE,VDLE,VKE,VFE)
C----- PRINT ELEMENT REACTIONS
    IF(M.GE.2) WRITE(MP,2000) IEL,(VFE(I),I=1,1D0)
2000 FORMAT(' REACTIONS (FE) , ELEMENT:',I5/(10X,1E12.5))
    IF(IRESD.NE.1) GO TO 20
C----- ASSEMBLE INTERNAL RESIDUALS
    CALL ASSEL(0,1,1D0,NSYM,KLOCE,KLD,VKE,VFE,VKGS,VKGD,VKGI,VRES)
20   IF(IREAC.NE.1) GO TO 60
C----- ASSEMBLE EXTERNAL REACTIONS
C     MODIFY TERMS IN KLOCE SUCH THAT PRESCRIBED D.O.F. ARE THE ONLY
C     ASSEMBLED ONES
    DO 50 ID=1,1D0
    IF(KLOCE(ID)) 30,50,40
30   KLOCE(ID)=KLOCE(ID)
    GO TO 50
40   KLOCE(ID)=0
50   CONTINUE
    CALL ASSEL(0,1,1D0,NSYM,KLOCE,KLD,VKE,VFE,VKGS,VKGD,VKGI,VREAC)
60   ITPE1=ITPE
    RETURN
    END
    SUBROUTINE ASSEL(IKG,IFG,1D0,NSYM,KLOCE,KLD,VKE,VFE,VKGS,
1      VKGD,VKGI,VFG)
C=====
C     TO ASSEMBLE AN ELEMENT MATRIX AND/OR VECTOR
C     (MATRIX SYMMETRICAL OR NOT)
C     INPUT
C       IKG      IF IKG.EQ.1 ASSEMBLE ELEMENT MATRIX KE
C       IFG      IF IFG.EQ.1 ASSEMBLE ELEMENT VECTOR FE
C       1D0      ELEMENT NUMBER OF D.O.F.
C       NSYM     0=SYMMETRIC PROBLEM, 1=UNSYMMETRIC PROBLEM
C       KLOCE    ELEMENT LOCALIZATION VECTOR
C       KLD      CUMULATIVE COLUMN HEIGHTS OF KG
C       VKE      ELEMENT MATRIX KE (FULL OR UPPER TRIANGLE BY
C                  DESCENDING COLUMN)
C       VFE      ELEMENT VECTOR FE
C     OUTPUT
C       VKGS,VKGD,VKI    GLOBAL MATRIX (SKYLINES)
C                      (SYMMETRIC OR NOT)
C       VFG      GLOBAL LOAD VECTOR
C=====
IMPLICIT REAL*8(A-H,O-Z)

```

```

DIMENSION KLOCE(*),KLD(*),VKE(*),VFE(*),VKGS(*),VKGD(*),
1 VKGI(*),VFG(*)
C-----
C----- ASSEMBLE ELEMENT MATRIX
IF(IKG.NE.1) GO TO 100
IEQO=IDLE
IEQI=1
C----- FOR EACH COLUMN OF KE
DO 90 JD=1, IDLE
IF(NSYM.NE.1) IEQO=JD
JL=KLOCE(JD)
IF(JL) 90,90,10
10 IO=KLD(JL+1)
IEQ=IEQI
IQ=1
C----- FOR EACH ROW OF KE
DO 80 ID=1, IDLE
IL=KLOCE(ID)
IF(NSYM.EQ.1) GO TO 30
IF(ID-JD) 30,20,20
20 IQ=ID
30 IF(IL) 80,80,40
40 IJ=JL-IL
IF(IJ) 70,50,60
C----- DIAGONAL TERMS OF KG
50 VKGD(IL)=VKGD(IL)+VKE(IEQ)
GO TO 80
C----- UPPER TRIANGLE TERMS OF KG
60 I=IO-IJ
VKGS(I)=VKGS(I)+VKE(IEQ)
GO TO 80
C----- LOWER TRIANGLE TERMS OF KG
70 IF(NSYM.NE.1) GO TO 80
I=KLD(IL+1)+IJ
VKGI(I)=VKGI(I)+VKE(IEQ)
80 IEQ=IEQ+IQ
90 IEQI=IEQI+IEQO
C----- ASSEMBLE ELEMENT LOAD VECTOR
100 IF(IFG.NE.1) GO TO 130
DO 120 ID=1, IDLE
IL=KLOCE(ID)
IF(IL) 120,120,110
110 VFG(IL)=VFG(IL)+VFE(ID)
120 CONTINUE
130 RETURN
END
SUBROUTINE MODFG(IDLE,NSYM,KLOCE,VDIMP,VKE,VFG)
C=====
C TO MODIFY VECTOR FG TO TAKE INTO ACCOUNT OF PRESCRIBED NON ZERO
C D.O.F. FOR A GIVEN ELEMENT
C INPUT
C           IDLE      ELEMENT NUMBER OF D.O.F.
C           NSYM     0=SYMMETRIC PROBLEM, 1=NON SYMMETRIC PROBLEM
C           KLOCE    ELEMENT LOCALIZATION VECTOR

```

```

C      VDIMP  VALUES OF PRESCRIBED D.O.F.
C      VKE    ELEMENT MATRIX (FULL OR UPPER TRIANGLE
C              BY DESCENDING COLUMNS)
C      OUTPUT
C      VFG    GLOBAL LOAD VECTOR
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION KLOCE(*),VDIMP(*),VKE(*),VFG(*)
DATA ZERO/0.0/DO/
C-----
IEQ0=IDLE
IEQ1=1
C----- FOR EACH ROW OF ELEMENT MATRIX
DO 50 JD=1,IDLE
IF(NSYM.NE.1) IEQ0=JD
IEQ=IEQ1
JL=KLOCE(JD)
IQ=1
IF(JL) 10,50,50
10   JL=-JL
DIMP=VDIMP(JL)
IF(DIMP.EQ.ZERO) GO TO 50
C----- FOR EACH COLUMN OF ELEMENT MATRIX
DO 40 ID=1,IDLE
IL=KLOCE(ID)
IF(NSYM.EQ.1) GO TO 30
IF(ID-JD) 30,20,20
20   IQ=ID
30   IF(IL.GT.0) VFG(IL)=VFG(IL)-VKE(IEQ)*DIMP
40   IEQ1=IEQ+IQ
50   IEQ1=IEQ1+IEQ0
      RETURN
      END
      SUBROUTINE PRPVTS(VKGD)
C=====
C      TO EVALUATE AND TO PRINT THE PIVOTS AND DETERMINANT OF MATRIX KG
C=====
IMPLICIT REAL*8(A-H,O-Z)
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
DIMENSION VKGD(*)
DATA UN/1./DO/,GROS/1.D38/
ABS(X)=DABS(X)
C-----
X1=GROS
X2=GROS
DET=UN
IDET=0
C----- PRINT PIVOTS OF MATRIX KG
IF(M.GE.2) WRITE(MP,2000) (VKGD(I),I=1,NEQ)
2000 FORMAT(' GLOBAL MATRIX PIVOTS'/(1X,10E12.5))
DO 50 I=1,NEQ
C----- ABSOLUTE VALUE OF MINIMUM PIVOT
X=ABS(VKGD(I))

```

```

IF(X.GT.X1) GO TO 10
X1=X
I1=I
C----- ALGEBRAIC VALUE OF MINIMUM PIVOT
X=VKGD(I)
10 IF(X.GT.X2) GO TO 20
X2=X
I2=I
C----- DETERMINANT (BOUNDS : 10 EXPONENT + OR - 10)
20 DET=DET*VKGD(I)
30 DET1=ABS(DET)
IF(DET1.LT.1.D10) GO TO 40
DET=DET*T.D-10
IDET=IDET+10
40 IF(DET1.GT.1.D-10) GO TO 50
DET=DET*I1.D10
IDET=IDET-10
GO TO 30
50 CONTINUE
C----- OUTPUT
WRITE(MP,2010) X1,I1,X2,I2,DET,IDET
2010 FORMAT(/15X,'ABSOLUTE VALUE OF MINIMUM PIVOT      =',E12.5,' EQUATION
1:',I5   /29X,           'ALGEBRAIC VALUE=',E12.5,' EQUATION:',,
2  I5   /29X,           'DETERMINANT      =',E12.5,' * 10 ** ',,
3  I5/)           RETURN
END
SUBROUTINE PRSOL(KDLNC,VCORG,VDIMP,KNEQ,VFG)
C=====
C     TO PRINT THE SOLUTION
C=====
IMPLICIT REAL*8(A-H,O-Z)
CHARACTER*4 RF,RL,FX
COMMON/COOR/NDIM,NNT,NDLN,NDLT
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
DIMENSION V(10),FX(10)
DIMENSION VDIMP(*),KDLNC(*),VCORG(*),KNEQ(*),VFG(*)
DATA RF/' * '/,RL/'   ',ZERO/0.D0/
C-----
X2=ZERO
X3=ZERO
WRITE(MP,2000)
2000 FORMAT('/ NODES',4X,'X',11X,'Y',11X,'Z',10X,'DEGREES OF FREEDOM (*
1 = PRESCRIBED) /')
I2=0
DO 50 IN=1,NNT
I1=I2+1
I2=I2+NDIM
ID1=KDLNC(IN)+1
ID2=KDLNC(IN+1)
ID=ID2-ID1+1
IF(ID2.LT.ID1) GO TO 50
X1=VCORG(I1)
IF(NDIM.GE.2) X2=VCORG(I1+1)

```

```

IF(NDIM.GE.3) X3=VCORG(I1+2)
J=ID1
DO 40 I=1,ID
JJ=KNEQ(J)
IF(JJ) 10,20,30
10 V(I)=VDIMP(-JJ)
FX(I)=RF
GO TO 40
20 V(I)=ZERO
FX(I)=RF
GO TO 40
30 V(I)=VFG(JJ)
FX(I)=RL
40 J=J+1
WRITE(MP,2010) IN,X1,X2,X3,(V(II),FX(II),II=1,ID)
2010 FORMAT(1X,I5,3E12.5,5X,5(E12.5,A4)/47X,5(E12.5,A4))
50 CONTINUE
RETURN
END
SUBROUTINE DLELM(KLOCE,VDLG,VDIMP,VDLE)
C=====
C TO GENERATE ELEMENT D.O.F.
C=====
IMPLICIT REAL*8(A-H,O-Z)
COMMON/RGDTIEL,IKEL,ITPE,ITPE1,ISDE,IMODEL,IFEL,IDL,E,INEL,
1 IPG,ICODE,IMATD,INSE,INPE,IDL0,INEL0,IPGO,IRES
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
DIMENSION KLOCE(*),VDLG(*),VDIMP(*),VDLE(*)
DATA ZERO/0.D0/
C-----
DO 40 ID=1,IDL
IL=KLOCE(ID)
IF(IL) 10,20,30
10 VDLE(ID)=VDIMP(-IL)
GO TO 40
20 VDLE(ID)=ZERO
GO TO 40
30 VDLE(ID)=VDLG(IL)
40 CONTINUE
IF(M.GE.2) WRITE(MP,2000) IEL,(VDLE(ID),ID=1,IDL)
2000 FORMAT(' DEGREES OF FREEDOM OF ELEMENT ',I5/(1X,10E12.5))
RETURN
END
SUBROUTINE SOL(VKGS,VKGD,VKGI,VFG,KLD,NEQ,MP,IFAC,ISOL,NSYM,ENERG)
C=====
C TO SOLVE A LINEAR SYSTEM (SYMMETRICAL OR NOT).
C THE MATRIX IS STORED IN CORE BY SKYLINES IN ARRAYS
C VKGS,VKGD,VKGI
C INPUT
C VKGS,VKGD,VKGI      SYSTEM MATRIX : UPPER, DIAGONAL AND
C                           LOWER PARTS
C VFG                  SECOND MEMBER
C KLD                  ADDRESSES OF COLUMN TOP TERMS
C NEQ                  NUMBER OF EQUATIONS

```

```

C      MP          OUTPUT DEVICE NUMBER
C      IFAC        IF IFAC.EQ.1 TRIANGULARIZE THE MATRIX
C      ISOL        IF ISOL.EQ.1 COMPUTE THE SOLUTION FROM
C                  TRIANGULARIZED MATRIX
C      NSYM        INDEX FOR NONSYMMETRIC PROBLEM
C      OUTPUT      VKGS,VKGD,VKGI   TRIANGULARIZED MATRIX (IF IFAC.EQ.1)
C                  VFG         SOLUTION (IF ISOL.EQ.1)
C                  ENERG      SYSTEM ENERGY (IF NSYM.EQ.0)
C=====
C      IMPLICIT REAL*8(A-H,O-Z)
C      DIMENSION VKGS(*),VKGD(*),VKGI(*),VFG(*),KLD(*)
C      DATA ZERO/0.0,D0/
C-----
C      IK=1
C      IF(VKGD(1).NE.ZERO) GO TO 10
C      WRITE(MP,2000) IK
C      STOP
C      10 ENERG=ZERO
C----- FOR EACH COLUMN IK EO BE MODIFIED
C      JHK=1
C      DO 100 IK=2,NEQ
C----- ADDRESS OF THE NEXT COLUMN TOP TERM IK+1
C      JHK1=KLD(IK+1)
C----- HEIGHT OF COLUMN IK (INCLUDE UPPER AND DIAGONAL TERMS)
C      LHK=JHK1-JHK
C      LHK1=LHK-1
C----- ROW OF FIRST TERM TO BE MODIFIED IN COLUMN IK
C      IMIN=IK-LHK1
C      IMIN1=IMIN-1
C----- ROW OF LAST TERM TO BE MODIFIED IN COLUMN IK
C      IMAX=IK-1
C      IF(LHK1.LT.0) GO TO 100
C      IF(IFAC.NE.1) GO TO 90
C      IF(NSYM.EQ.1) VKGI(JHK)=VKGI(JHK)/VKGD(IMIN1)
C      IF(LHK1.EQ.0) GO TO 40
C----- MODIFY NON-DIAGONAL TERM IN COLUMN IK
C      JCK=JHK+1
C      JHJ=KLD(IMIN)
C----- FOR EACH TERM LOCATED AT JCK AND CORRESPONDING TO COLUMN IJ
C      DO 30 IJ=IMIN,IMAX
C      JHJ1=KLD(IJ+1)
C----- NUMBER OF MODIFICATIVE TERMS FOR COEFFICIENT LOCATED AT JCK
C      IC=MIN0(JCK-JHK,JHK1-JHJ)
C      IF(IC.LE.0.AND.NSYM.EQ.0) GO TO 20
C      C1=ZERO
C      IF(IC.LE.0) GO TO 17
C      J1=JHK1-IC
C      J2=JCK-IC
C      IF(NSYM.EQ.1) GO TO 15
C      VKGS(JCK)=VKGS(JCK)-SCAL(VKGS(J1),VKGS(J2),IC)
C      GO TO 20
C      15 VKGS(JCK)=VKGS(JCK)-SCAL(VKGI(J1),VKGS(J2),IC)
C      C1=SCAL(VKGS(J1),VKGI(J2),IC)

```

```

17      VKGI(JCK)=(VKGI(JCK)-C1)/VKGD(IJ)
20      JCK=JCK+1
30      JHJ=JHJ1
C----- MODIFY DIAGONAL TERM
40      JCK=JHK
        CDIAG=ZERO
        DO 70 IJ=IMIN1,IMAX
          C1=VKGS(JCK)
          IF(NSYM.EQ.1) GO TO 50
          C2=C1/VKGD(IJ)
          VKGS(JCK)=C2
          GO TO 60
50      C2=VKGI(JCK)
60      CDIAG=CDIAG+C1*C2
70      JCK=JCK+1
        VKGD(IK)=VKGD(IK)-CDIAG
        IF(VKGD(IK)) 90,80,90
80      WRITE(MP,2000) IK
2000  FORMAT(' *** ERROR,ZERO PIVOT EQUATION ',I5)
      STOP
C----- SOLVE LOWER TRIANGULAR SYSTEM
90      IF(ISOL.NE.1) GO TO 100
        IF(NSYM.NE.1) VFG(IK)=VFG(IK)-SCAL(VKGS(JHK),VFG(IMIN1),LHK)
        IF(NSYM.EQ.1) VFG(IK)=VFG(IK)-SCAL(VKGI(JHK),VFG(IMIN1),LHK)
100    JHK=JHK1
        IF(ISOL.NE.1) RETURN
C----- SOLVE DIAGONAL SYSTEM
        IF(NSYM.EQ.1) GO TO 120
        DO 110 IK=1,NEQ
          C1=VKGD(IK)
          C2=VFG(IK)/C1
          VFG(IK)=C2
110    ENERG=ENERG+C1*C2*C2
C----- SOLVE DIAGONAL SYSTEM
120    IK=NEQ+1
        JHK1=KLD(IK)
130    IK=IK-1
        IF(NSYM.EQ.1) VFG(IK)=VFG(IK)/VKGD(IK)
        IF(IK.EQ.1) RETURN
        C1=VFG(IK)
        JHK=KLD(IK)
        JBK=JHK1-1
        IF(JHK.GT.JBK) GO TO 150
        IJ=IK-JBK+JHK-1
        DO 140 JCK=JHK,JBK
          VFG(IJ)=VFG(IJ)-VKGS(JCK)*C1
140    IJ=IJ+1
150    JHK1=JHK
        GO TO 130
        END
        FUNCTION SCAL(X,Y,N)
C=====
C      INNER PRODUCT OF VECTORS X AND Y OF LENGTH N
C      (FUNCTION TO BE WRITTEN EVENTUALLY IN ASSEMBLER)

```

```

C=====
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION X(*),Y(*)
      DATA ZERO/0.0/D0/
C-----
      SCAL=ZERO
      DO 10 I=1,N
10    SCAL=SCAL+X(I)*Y(I)
      RETURN
      END
      SUBROUTINE BLNLIN
C=====
C      TO CALL BLOCK 'NLIN'
C      TO SOLVE A STEADY NON LINEAR PROBLEM
C=====
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*4 TBL
      COMMON/COOR/NDIM,NNT,NDLN,NDLT
      COMMON/PREL/NGPE,NPRE,NPRM,NMNL,NHIS
      COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,LAG,NIDENT,NPG,NSEM,
      1 NPEM,NELS
      COMMON/ASSE/NEQ,NSYM,NKG,NKE,NLDE
      COMMON/NLIN/EPSDL,OMEGA,DPAS,NPAS,NITER,ITER,IMETH,ILAG,INLN
      COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
      COMMON/LOCA/LCORG,LDLNC,LNEQ,LDIMP,LCODE,LINTF,LREG,LLD,LLOC,
      1 LCORE,LNE,LFREE,LNPEG,LSDEG,LDLE,LKE,LFE,LKGS,LKGD,LKGI,LFG,
      2 LRES,LDLG,LNELS,LEB,LPB
      COMMON/SEQU/NSEQ,NSEQO,ISEQ,IASSEL,IBAR
      COMMON VA(20000)
      DIMENSION TBL(9)
      DATA TBL/'KGS ','KGD ','KGI ','FG ','KE ','FE ','RES ','DLE ',
      1 'DLG '/
C-----
      ILAG=LAG
      INLN=1
      NSEQ=NSEQ+1
      NSEQO=NSEQO+1
      WRITE(*,2000) M
      WRITE(MP,2000) M
2000 FORMAT(//' NON LINEAR SOLUTION (M=',I2,')'/1X,23('='))
      IF(NSEQO.NE.1) GO TO 10
      IF(NMNL.EQ.0) GO TO 10
C----- OPEN DIRECT ACCESS FILE M3 (ELASTO-PLASTIC MATRICES)
      LD=16*8
      OPEN(M3,FILE='M3.DAT',STATUS='NEW',ACCESS='DIRECT',RECL=LD)
C----- OPEN DIRECT ACCESS FILE M9 (STIFFNESS IDENTIFIER DATA)
      LT=NPEM*4
      OPEN(M9,FILE='M9.DAT',STATUS='NEW',ACCESS='DIRECT',RECL=LT)
C----- OPEN DIRECT ACCESS FILE MH (PATH HISTORY DATA)
      IF(NHIS.EQ.0) GO TO 10
      LH=NPEM*NHIS*8
      OPEN(MH,FILE='MH.DAT',STATUS='NEW',ACCESS='DIRECT',RECL=LH)
10    IF(NSEQ.NE.1) GO TO 20
C----- OPEN DIRECT ACCESS FILE MS (STRESSES, STRAINS AND PWPS)

```

```

LS=NSEM*8
OPEN(MS,FILE='MS.DAT',STATUS='NEW',ACCESS='DIRECT',RECL=LS)
C----- TO ALLOCATE SPACE
  IF(LKGS.EQ.1) CALL ESPACE(NKG,1,TBL(1),LKGS)
  IF(LKGD.EQ.1) CALL ESPACE(NEQ,1,TBL(2),LKGD)
  IF(NSYM.EQ.1.AND.LXGI.EQ.1) CALL ESPACE(NKG,1,TBL(3),LXGI)
  IF(LFG.EQ.1) CALL ESPACE(NEQ,1,TBL(4),LFG)
  IF(LKE.EQ.1) CALL ESPACE(NKE,1,TBL(5),LKE)
  IF(LFE.EQ.1) CALL ESPACE(NDLE,1,TBL(6),LFE)
  IF(LRES.EQ.1) CALL ESPACE(NDLT+NEQ,1,TBL(7),LRES)
  IF(LDLE.EQ.1) CALL ESPACE(NDLE,1,TBL(8),LDLE)
  IF(LDLG.EQ.1) CALL ESPACE(NEQ,1,TBL(9),LDLG)
C----- TO EXECUTE THE BLOCK
20  CALL EXNLIN(VA(LLD),VA(LDIMP),VA(LLOCE),VA(LCORE),VA(LCODE),
   1 VA(LINTP),VA(LPREG),VA(LPREE),VA(LNE),VA(LKE),VA(LFE),
   2 VA(LKGS),VA(LKGD),VA(LXGI),VA(LFG),VA(LCORG),VA(LDLNC),
   3 VA(LNEQ),VA(LNPEG),VA(LRES),VA(LSDEG),VA(LDLE),VA(LDLG))
  RETURN
  END
  SUBROUTINE EXNLIN(KLD,VDIMP,KLOCE,VCore,KCode,KINTF,VPREG,VPREE,
  1 KNE,VKE,VFE,VKGS,VKGD,VKGI,VFG,VCORG,KDLNC,KNEQ,KNPEG,VRES,
  2 KSDEG,VDLE,VDLG)
C=====
C----- TO EXECUTE BLOCK 'NLIN'
C----- TO SOLVE A STEADY NON LINEAR PROBLEM BY NEWTON-RAPHSON METHOD
C----- IMETH.EQ.1 COMPUTE K AT EACH ITERATION
C----- IMETH.EQ.2 K IS CONSTANT AT EACH ITERATION
C----- IMETH.EQ.3 RECOMPUTE K AT THE BEGINNING OF EACH STEP
C----- IMETH.EQ.4 K IS LINEAR AT THE FIRST ITERATION(DEFAULT)
C=====
IMPLICIT REAL*8(A-H,O-Z)
COMMON/COOR/NDIM,NNT,NDLN,NDLT
COMMON/PREL/NGPE,NPRE,NPRM,NMNL,NHIS
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1 NPEM,NELS
COMMON/NLIN/EPSDL,OMEGA,DPAS,NPAS,NITER,ITER,IMETH,ILAG,INLN
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
COMMON/SEQU/NSEQ,NSEQO,ISEQ,IASSEL,IBAR
DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCore(*),KCode(*),KINTF(*),
1 VPREG(*),VPREE(*),KNE(*),VKE(*),VFE(*),VKGS(*),VKGD(*),
2 VKGI(*),VFG(*),VCORG(*),KDLNC(*),KNEQ(*),KNPEG(*),VRES(*),
3 KSDEG(*),VDLE(*),VDLG(*)
  DATA ZERO/0.D0/,UN/1.D0/
C-----
C----- INITIALIZE FILE MS
  IF(NSEQ.EQ.1) CALL INITMS(NELT,NSEM,MS)
  IF(NSEQO.NE.1) GO TO 10
C----- INITIALIZE FILES M9 AND MH IF DEAL WITH NONLINEAR MATERIAL
  IF(NMNL.EQ.1) CALL INITM9(M9,NELT,NPEM)
  IF(NHIS.GT.0) CALL INITMH(VPREG,VPREE,KNPEG,KCode,MS,MH,NPRE,
1 NPRM,NELT)
10  DPASO=ZERO
  XPAS=ZERO

```

```

IPAS=0
C----- READ A CARD DEFINING A SET OF IDENTICAL STEPS
      READ(MR,1000) DPAS,I1,I2,I3,X1,X2,IELS,IEQL
1000  FORMAT(F10.0,3I5,2F10.0,2I5)
      IF(I1.GT.0) NPAS=I1
      IF(I2.GT.0) NITER=I2
      IF(I3.GT.0) IMETH=I3
      IF(X1.GT.ZERO) EPSDL=X1
      IF(X2.GT.ZERO) OMEGA=X2
C----- LOOP OVER ALL STEPS
      DO 130 IP=1,NPAS
      IPAS=IPAS+1
      WRITE(*,1900) IPAS
1900  FORMAT(' INCREMENT NUMBER = ',I5)
      XPAS=XPAS+DPAS
      WRITE(MP,2000) IPAS,DPAS,XPAS,NITER,IMETH,EPSDL,OMEGA
2000  FORMAT(/1X,13(' -'),'STEP NUMBER (IPAS):',I5//,
     1          14X,'INCREMENT           (DPAS)=',E12.5/
     2          14X,'TOTAL LEVEL        (XPAS)=',E12.5/
     3          14X,'NUMBER OF ITERATIONS (NITER)=',I12/
     4          14X,'METHOD NUMBER       (IMETH)=',I12/
     5          14X,'TOLERANCE          (EPSDL)=',E12.5/
     6          14X,'OVER RELAXATION FACTOR (OMEGA)=',E12.5)
C----- INCREMENT THE LOADS AND STORE IN THE RESIDUAL VECTOR
      CALL MAJ(DPAS,ZERO,NEQ,VFG,VRES)
C----- UPDATE TOTAL-LOAD LOAD VECTOR
      DO 20 I=1,NEQ
20    VRES(NDLT+I)=VRES(NDLT+I)+VRES(I)
C----- LOOP OVER EQUILIBRIUM ITERATIONS
      DO 110 ITER=1,NITER
      IF(IMETH.GT.4) WRITE(MP,2010) IMETH
2010  FORMAT(' ** ERROR,METHOD:',I3,' UNKNOWN')
      IF(IMETH.GT.4) STOP
C----- DECIDE IF GLOBAL MATRIX IS TO BE ASSEMBLED
      IKT=0
      IF(IMETH.EQ.1.OR.IMETH.EQ.4) IKT=1
      IF(IMETH.EQ.2.AND.IPAS.EQ.1.AND.ITER.EQ.1) IKT=1
      IF(IMETH.EQ.3.AND.ITER.EQ.1) IKT=1
      IF(IKT.EQ.0) GO TO 60
C----- INITIALIZE GLOBAL MATRIX TO ZERO IF IT IS TO BE ASSEMBLED
      IF(NSEQ.EQ.1.AND.IPAS.EQ.1.AND.ITER.EQ.1) GO TO 50
      DO 30 I=1,NKG
      VKGS(I)=ZERO
30    IF(NSYM.EQ.1) VKGI(I)=ZERO
      DO 40 I=1,NEQ
40    VKGD(I)=ZERO
C----- ASSEMBLE KG
50    CALL ASKG(KLD,VDIMP,KLOCE,VCORE,KCODE,KINTF,VPREG,VPREE,KNE,
     1 VKE,VFE,VKGS,VKGD,VKGI,VFG,KNPEG,KSDEG,VDLE,VDLG)
C----- SOVLE
60    CALL SOL(VKGS,VKGD,VKGI,VRES,KLD,NEQ,MP,IKT,1,NSYM,ENERG)
      IF(IKT.EQ.1.AND.M.GT.1) CALL PRPVTS(VKGD)
C----- UPDATE STRESSES, STRAINS AND PWPS
      CALL ASGRAD(KLD,VDIMP,KLOCE,VCORE,KCODE,KINTF,VPREG,VPREE,KNE,

```

```

1 VKE,VFE,VKGS,VKGD,VKGI,VRES,KNPEG,KSDEG,VDLE,VDLG,NRES)
C----- UPDATE THE SOLUTION
    CALL MAJ(OMEGA,UN,NEQ,VRES,VDLG)
C----- COMPUTE THE NORM
    CALL NORME(NEQ,VRES,VDLG,XNORM)
    IF(M.GT.0) WRITE(MP,2020) ITER,XNORM
2020 FORMAT(5X,'ITERATION (ITER):',I3,' NORM (XNORM)=',E12.5)
    IF(M.GE.2) CALL PRSOL(KDLNC,VCORG,VDIMP,KNEQ,VDLG)
C----- UPDATE STRESSES AND STRAINS AND PWPS
    IF(NRES.EQ.0) GO TO 120
    IF(XNORM.LE.EPSDL) GO TO 120
C----- ASSEMBLE RESIDUAL VECTOR
    DO 70 I=1,NEQ
70   VRES(I)=ZERO
    CALL ASRESD(1,0,KLD,VDIMP,KLOCE,VCORE,VPREG,VPREE,KNE,VKE,VFE,
1   VKGS,VKGD,VKGI,VFG,KNPEG,KSDEG,VDLE,VDLG,VRES,VRES(NEQ+1))
    DO 80 I=1,NEQ
80   VRES(I)=VRES(NDLT+I)-VRES(I)
110  CONTINUE
    ITER=NITER
C----- END OF STEP
120  DPASO=DPAS
    WRITE(MP,2030) ITER,NITER
2030 FORMAT(/10X,I4,' PERFORMED ITERATIONS OVER',I4/)
    IF(M.LT.2) CALL PRSOL(KDLNC,VCORG,VDIMP,KNEQ,VDLG)
    IF(IELS.EQ.1) CALL ELSOL(KLD,VDIMP,KLOCE,VCORE,VPREE,KNE,VKE,
1   VFE,VKGS,VKGD,VKGI,VDLE,VDLG,VRES,IPAS,NLAG,IEQL)
130  CONTINUE
C----- FOR THE PROBLEM WITHOUT CONSTRUCTION SEQUENCES
C----- EVALUATE AND PRINT EQUILIBRIUM RESIDUAL VECTOR
    IF(ISEQ.NE.0) RETURN
C----- SAVE VECTOR VFG IN VRES AND CHANGE SIGN
    DO 140 I=1,NEQ
140  VRES(I)=-VRES(NDLT+I)
C----- ASSEMBLE THE RESIDUALS
    CALL ASRESD(1,1,KLD,VDIMP,KLOCE,VCORE,VPREG,VPREE,KNE,VKE,VFE,
1   VKGS,VKGD,VKGI,VFG,KNPEG,KSDEG,VDLE,VDLG,VRES,VRES(NEQ+1))
C----- PRINT THE RESIDUALS
    WRITE(MP,2040)
2040 FORMAT(//' EQUILIBRIUM RESIDUALS AND REACTIONS'//)
    CALL PRSOL(KDLNC,VCORG,VRES(NEQ+1),KNEQ,VRES)
    RETURN
    END
    SUBROUTINE INITMS(NELT,NSEM,MS)
C=====
C----- INITIALIZE FILE MS (DATA FOR STRESSES, STRAINS AND PWPS)
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION VSE(81)
DO 10 I=1,NSEM
10  VSE(I)=0.D0
DO 20 IEL=1,NELT
20  WRITE(MS,REC=IEL) (VSE(I),I=1,NSEM)
    RETURN

```

```

END
SUBROUTINE INITM9(M9,NELT,NPEM)
C=====
C   INITIALIZE FILE M9 (STIFFNESS IDENTIFIER DATA)
C=====
DIMENSION MT(9)
DO 10 I=1,NPEM
10  MT(I)=1
DO 20 IEL=1,NELT
20  WRITE(M9,REC=IEL) (MT(I),I=1,NPEM)
RETURN
END
SUBROUTINE INITMH(VPREG,VPREE,KNPEG,KCODE,MS,MH,NPRE,NPRM,NELT)
C=====
C   INITIALIZE FILE MH (PATH HISTORY DATA)
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION VPREG(*),VPREE(*),KNPEG(*),KCODE(*)
DO 20 IEL=1,NELT
IGPE=KNPEG(IEL)
IMODEL=KCODE(IGPE)
IF(IMODEL.NE.4) GO TO 20
IC=(IGPE-1)*(NPRE+NPRM)
DO 10 I=1,NPRE+NPRM
10  VPREE(I)=VPREG(IC+I)
IF(IMODEL.EQ.4) CALL CAPI(VPREE,NPRE,IEL,MS,MH)
20  CONTINUE
RETURN
END
SUBROUTINE MAJ(X1,X2,N,V1,V2)
C=====
C   EXECUTE THE VECTOR OPERATION: V2=X1*V1 + X2*V2
C   X1,X2:SCALARS  V1,V2:VECTORS
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION V1(*),V2(*)
C-----
DO 10 I=1,N
10  V2(I)=X1*V1(I)+X2*V2(I)
RETURN
END
SUBROUTINE NORME(N,VDEL,V,XNORM)
C=====
C   COMPUTE THE LENGTHS RATIO OF VECTORS VDEL AND V
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION VDEL(*),V(*)
DATA ZERO/0.D0/,UN/1.D0/,FAC/1.D-3/
SQRT(X)=DSQRT(X)
C-----
C1=ZERO
C2=ZERO
DO 10 I=1,N
C1=C1+VDEL(I)*VDEL(I)

```

```

10  C2=C2+V(I)*V(I)
    C=C1*FAC
    IF(C2.LE.C) C2=UN
    XNORM=SQRT(C1/C2)
    RETURN
    END
    SUBROUTINE ELSOL(KLD,VDIMP,KLOCE,VCORE,VPREE,KNE,VKE,VFE,VKGS,
    1 VKGD,VKG,I,VDLE,VDLG,VRES,IPAS,NLAG,IEQL)
C=====
C      TO PRINT SOLUTIONS FOR SPECIFIED ELEMENTS FOR EACH INCREMENT
C      IN A NONLINEAR ANALYSIS AND TO FIND EQUIVALENT NODAL FORCES
C      DUE TO THE STRESSES AT GAUSS POINTS IF IEQL .EQ. 1
C=====
    IMPLICIT REAL*8(A-H,O-Z)
    COMMON/ASSE/NEQ,NSYM,NKG,NKE,NLDE
    COMMON/RGDT/IEL,IKEL,ITPE,ITPE1,ISDE,IMODEL,IFEL,IDLE,ICE,INEL,
    1 IPG,ICODE,IMATD,INSE,INPE,IDELO,INEL0,IPGO,IRES
    COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
    DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCORE(*),VPREE(*),KNE(*),
    1 VKE(*),VFE(*),VKGS(*),VKGD(*),VKG(I,*),VDLE(*),VDLG(*),VRES(*)
    DIMENSION NSOL(50)
C-----
    IF(IPAS.NE.1) GO TO 20
C----- READ NUMBER OF ELEMENTS
    READ(MR,1000) NELS
1000 FORMAT(I5)
C----- READ ELEMENT NUMBER CARD
    N=1
    I1=1
10   I2=MIN0(N*16,NELS)
    READ(MR,1500) (NSOL(I),I=I1,I2)
1500 FORMAT(16I5)
    IF(NELS.EQ.I2) GO TO 20
    N=N+1
    I1=I1+16
    GO TO 10
C----- INITIALIZE VRES
20   DO 30 I=1,NEQ
30   VRES(I)=0.D0
C----- LOOP OVER THE SPECIFIED ELEMENTS
    DO 50 IEL=1,NELS
    IEL=NSOL(IE)
C----- READ THE ELEMENT
    CALL RDELEM(ME,KLOCE,VCORE,KNE)
C----- EVALUATE INTERPOLATION FUNCTIONS IF REQUIRED
    IF(ITPE.EQ.ITPE1) GO TO 40
    ICODE=2
    CALL ELEMLB(VCORE,VPREE,VDLE,VKE,VFE)
C----- PRINT STRESSES AND STRAINS
40   ICODE=10
    CALL ELEMLB(VCORE,VPREE,VDLE,VKE,VFE)
    IF(IEQL.EQ.0) GO TO 50
C----- EVALUATE EQUIVALENT NODAL FORCES DUE TO STRESSES AT G.P.
    IF(NLAG.GT.0) CALL DLELM(KLOCE,VDLG,VDIMP,VDLE)

```

```

ICODE=9
CALL ELEMELB(VCORE,VPREE,VDLE,VKE,VFE)
C----- ASSEMBLE
      CALL ASSEL(0,1,IDL,NSYM,KLOCE,KLD,VKE,VFE,VKGS,VKGD,VKGI,VRES)
50   ITPE1=ITPE
C----- PRINT EQUIVALENT NODAL FORCES
      IF(IEQL.EQ.1) WRITE(MP,2000) (VRES(I),I=1,NEQ)
2000 FORMAT(/' GLOBAL LOAD VECTOR FOR SPECIFIED ELEMENTS'/(5X,6E12.5))
      RETURN
      END
      SUBROUTINE BLPRNT
C=====
C TO CALL BLOCK 'PRNT'
C TO PRINT SOLUTIONS FOR SPECIFIED ELEMENTS
C=====
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*4 TBL
      COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1     NPEM,NELS
      COMMON/LOCA/LCORG,LDCNC,LNEQ,LDIMP,LCODE,LINTF,LPREG,LLD,LLOCE,
1     LCORE,LNE,LPREE,LNPEG,LSDEG,LDE,LKE,LFE,LKGS,LKGD,LKGI,LFG,
2     LRES,LDLN,LNELS,LEB,LPB
      COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
      COMMON VA(20000)
      DATA TBL/'NELS'/
C-----
C----- READ NUMBER OF ELEMENTS
      READ(MR,1000) NELS
1000 FORMAT(15)
      WRITE(MP,2000) M,NELS
2000 FORMAT(// OUTPUT OF SOLUTIONS FOR SPECIFIED ELEMENTS (M=',I2,
1   ')')// ',50(''=')/15X,'NUMBER OF SPECIFIED ELEMENTS (NP)=' ,I5)
      CALL ESPACE(NELS,0,TBL,LNELS)
      CALL EXPRNT(VA(LLOCE),VA(LCORE),VA(LPREE),VA(LNE),VA(LDLE),
1     VA(LKE),VA(LFE),VA(LSDEG),VA(LNELS))
      CALL DSPACE(LNELS,0,TBL)
      RETURN
      END
      SUBROUTINE EXPRNT(KLOCE,VCORE,VPREE,KNE,VDLE,VKE,VFE,KSDEG,KNELS)
C=====
C TO EXECUTE BLOCK 'PRNT'
C TO PRINT SOLUTIONS FOR SPECIFIED ELEMENTS
C=====
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1     NPEM,NELS
      COMMON/RGDT/IEL,IKEI,ITPE,ITPE1,ISDE,IMODEL,IFEL,IDL,ICE,INEL,
1     IPG,ICODE,IMATD,INSE,INPE,IDL0,INEL0,IPGO,IRES
      COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
      DIMENSION KLOCE(*),VCORE(*),VPREE(*),KNE(*),VDLE(*),VKE(*),
1     VFE(*),KSDEG(*),KNELS(*)
C-----
C----- READ AN ELEMENT PRINTOUT CARD
      KK=0

```

```

10    READ(MR,1000) IEL,IGEN,INCR
1000  FORMAT(3I5)
      IF(IEL.LE.0) GO TO 30
      DO 20 I=1,IGEN
      KK=KK+1
      KNELS(KK)=IEL
20    IEL=IEL+INCR
      GO TO 10
C----- LOOP OVER THE SPECIFIED ELEMENTS
30    DO 50 IE=1,NELS
      IEL=KNELS(IE)
C----- READ THE ELEMENT
      CALL RDELEM(ME,KLOCE,VCORE,KNE)
      ISDE=KSDEG(IEL)
C----- EVALUATE INTERPOLATION FUNCTION IF REQUIRED
      IF(ITPE.EQ.ITPE1) GO TO 40
      ICODE=2
      CALL ELEMLB(VCORE,VPREE,VDLE,VKE,VFE)
C----- PRINT STRESSES, STRAINS, PWPS AND PRINCIPAL STRESSES
40    ICODE=10
      CALL ELEMLB(VCORE,VPREE,VDLE,VKE,VFE)
50    ITPE1=ITPE
      RETURN
      END
      SUBROUTINE BLPLOT
C=====
C      TO CALL BLOCK 'PLOT'
C      TO CREATE DATA FILE M1 TO BE USED BY FENCON3.EXE (BASIC)
C=====
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/LOCA/LCORG,LDLNC,LNEQ,LDIMP,LCODE,LINTF,LPREG,LLD,LLOCE,
1     LCORE,LNE,LPREE,LNPEG,LSDEG,LDLE,LKE,LFE,LKGS,LKGD,LKGI,LFG,
2     LRES,LDLG,LNELS,LEB,LPB
      COMMON VA(20000)
C=====
      CALL EXPLOT(VA(LCORG),VA(LCORE),VA(LDLG),VA(LDLE),VA(LDIMP),
1     VA(LNPEG),VA(LLOCE),VA(LNE))
      RETURN
      END
      SUBROUTINE EXPLOT(VCORG,VCORE,VDLG,VDLE,VDIMP,KNPEG,KLOCE,KNE)
C=====
C      TO EXECUTE BLOCK 'PLOT'
C      TO CREATE DATA FILE M1
C=====
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/COOR/NDIM,NNT,NDLN,NDLT
      COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1     NPEM,NELS
      COMMON/RGDTIEL,IKEL,ITPE,ITPE1,ISDE,IMODEL,IFEL,IDL,E,ICE,INEL,
1     IPG,ICODE,IMATD,INSE,INPE,IDE,INELO,IPGO,IRES
      COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
      COMMON/SEQU/NSEQ,NSEQO,ISEQ,IASSEL,IBAR
      DIMENSION VCORG(*),VCORE(*),VDLG(*),VDLE(*),VDIMP(*),KNPEG(*),
1     KLOCE(*),KNE(*)

```

```
C-----  
C----- FIND MAXIMUM AND MINIMUM X'S AND Y'S  
    XMIN=0.D0  
    YMIN=0.D0  
    XMAX=0.D0  
    YMAX=0.D0  
    DO 10 I=1,NNT  
    X=VCORG(I*2-1)  
    IF(XMIN.GT.X) XMIN=X  
    IF(XMAX.LT.X) XMAX=X  
    Y=VCORG(I*2)  
    IF(YMIN.GT.Y) YMIN=Y  
10   IF(YMAX.LT.Y) YMAX=Y  
    WRITE(M1,2000) NELT  
2000 FORMAT(I5)  
    WRITE(M1,2010) XMIN,YMIN,XMAX,YMAX  
2010 FORMAT(4F10.2)  
    WRITE(M1,2000) ISEQ  
C----- LOOP OVER ELEMENTS  
    DO 20 IEL=1,NELT  
    WRITE(M1,2000) KNPEG(IEL)  
    CALL RDELEM(ME,KLOCE,VCORE,KNE)  
    WRITE(M1,2000) INEL  
C----- INPUT COORDINATES  
    WRITE(M1,2020) (VCORE(I),I=1,INEL*2)  
2020 FORMAT(8F10.2)  
C----- INPUT ELEMENT D.O.F.  
    CALL DLELM(KLOCE,VDLG,VDIMP,VDLE)  
    WRITE(M1,2020) (VDLE(I),I=1,INEL*2)  
20   CONTINUE  
    RETURN  
    END
```

```

SUBROUTINE BLIND
C=====
C      TO CALL BLOCK 'LIND'
C      TO ASSEMBLE AND TO SOLVE A LINEAR PROBLEM WHEN MATRIX KG IS
C      STORED BLOCKWISE ON DISK
C=====
IMPLICIT REAL*8(A-H,O-Z)
CHARACTER*4 TBL
COMMON/COOR/NDIM,NNT,NDLN,NDLT
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,LAG,NIDENT,NPG,NSEM,
1 NPEM,NELS
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
COMMON/ALLO/NVA,IVA,IVAMAX,NREEL,NTBL
COMMON/LOCA/LCORG,LDLNC,LNEQ,LDIMP,LCODE,LINTF,LPREG,LLD,LLOC,
1 LCORE,LNE,LPREE,LNPEG,LDLNG,LDLEG,LDE,LKE,LFE,LKGS,LKGD,LKGI,LFG,
2 LRES,LDLG,LNELS,LEB,LPB
COMMON/SEQU/NSEQ,NSEQO,ISEQ,IASSEL,IBAR
COMMON/NLIN/EPSDL,OMEGA,DPAS,NPAS,NITER,ITER,IMETH,ILAG,INLN
COMMON/DISK/NLBL,NBLM
COMMON VA(20000)
DIMENSION TBL(11)
DATA TBL/'KGS ','KGD ','KGI ','FG ','KE ','FE ','RES ','DLE ',
1 'DLG ','EB ','PB '/,DEUX/2.D0/,NBLMAX/100/
C-----
ILA=0
INLN=0
NSEQ=NSEQ+1
WRITE(MP,2000) M
2000 FORMAT(// ' ON DISK ASSEMBLAGE AND LINEAR SOLUTION (M=',I2,')!/
1 ' ',I2('='))
IF(NSEQ.GT.1) GO TO 40
C----- OPEN DIRECT ACCESS FILE MS (STRESSES, STRAINS & PWPS)
LS=NSEM*8
OPEN(MS,FILE='MS.DAT',STATUS='NEW',ACCESS='DIRECT',RECL=LS)
C----- OPEN SEQUENTIAL FILES, M4 AND M7 (STORE MATRIX KG)
OPEN(M4,FILE='M4.DAT',STATUS='NEW',FORM='UNFORMATTED')
OPEN(M7,FILE='M7.DAT',STATUS='NEW',FORM='UNFORMATTED')
C----- TO ALLOCATE SPACE
IF(LKGD.EQ.1) CALL ESPACE(NEQ,1,TBL(2),LKGD)
IF(LFG.EQ.1) CALL ESPACE(NEQ,1,TBL(4),LFG)
IF(LKE.EQ.1) CALL ESPACE(NKE,1,TBL(5),LKE)
IF(LFE.EQ.1) CALL ESPACE(NDLE,1,TBL(6),LFE)
IF(LRES.EQ.1) CALL ESPACE(NDLT+NEQ,1,TBL(7),LRES)
IF(LDLE.EQ.1) CALL ESPACE(NDLE,1,TBL(8),LDLE)
IF(LDLG.EQ.1) CALL ESPACE(NEQ,1,TBL(9),LDLG)
C----- FIND BLOCK LENGTH
NLBL=0
NBLM=0
I3=2
I2=1+NSYM
IF(NLBL.EQ.0) GO TO 10
IF(NBLM.EQ.0) NBLM=NKG/NLBL+2
GO TO 30

```

```

10    I1=NVA-IVA-(2*NBLMAX+2)/NREEL-1
     IF(I1.GE.(NKG*I2+2)) GO TO 20
C----- CASE WHERE MATRIX IS TO BE SEGMENTED
     NLBL=I1/(DEUX*I2)
     NBLM=NKG/NLBL+2
     GO TO 30
C----- CASE WHERE MATRIX IS IN CORE
20    NLBL=NKG
     NBLM=1
     I3=1
30    WRITE(MP,2010) NLBL,NBLM
2010  FORMAT(
1   15X,'BLOCKS LENGTH IN KG          (NLBL)=',I5/
2   15X,'MAX. NUMBER OF BLOCKS IN KG      =',I5)
     CALL ESPACE(NBLM+1,0,TBL(10),LEB)
     CALL ESPACE(NBLM,0,TBL(11),LPB)
     IF(LKGS.EQ.1) CALL ESPACE(NLBL*I3,1,TBL(1),LKGS)
     IF(NSYM.EQ.1.AND.LKGI.EQ.1) CALL ESPACE(NLBL*I3,1,TBL(3),LKGI)
40    CALL EXLIND(VA(LLD),VA(LDIMP),VA(LLOC),VA(LCORE),VA(LCODE),
1   VA(LINTF),VA(LPREG),VA(LPREE),VA(LNE),VA(LKE),VA(LFE),VA(LKGS),
2   VA(LKGD),VA(LKGI),VA(LFG),VA(LCORG),VA(LDLC),VA(LNEQ),
3   VA(LNPEG),VA(LRES),VA(LSDEG),VA(LDLE),VA(LDLG),VA(LEB),VA(LPB))
     RETURN
     END
     SUBROUTINE EXLIND(KLD,VDIMP,KLOCE,VCORE,KCODE,KINTF,VPREG,VPREE,
1   KNE,VKE,VFE,VKGS,VKGD,VKGI,VFG,VCORG,KDLNC,KNEQ,KNPEG,VRES,
2   KSDEG,VDLE,VDLG,KEB,KPB)
C=====
C----- TO EXECUTE BLOCK 'LIND'
C----- ASSEMBLE AND SOLVE A LINEAR PROBLEM WHEN MATRIX KG IS STORED
C----- BLOCKWISE ON DISK
C=====
IMPLICIT REAL*8(A-H,O-Z)
COMMON/COOR/NDIM,NNT,NDLN,NDLT
COMMON/COND/NCLT,NCLZ,NCLNZ
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1 NPEM,NELS
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
COMMON/SEQU/NSEQ,NSEQ0,ISEQ,IASSL,IBAR
COMMON/DISK/NLBL,NBLM
DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCORE(*),KCODE(*),KINTF(*),
1 VPREG(*),VPREE(*),KNE(*),VKE(*),VFE(*),VKGS(*),VKGD(*),VKGI(*),
2 VFG(*),VCORG(*),KDLNC(*),KNEQ(*),KNPEG(*),VRES(*),KSDEG(*),
3 VDLE(*),VDLG(*),KEB(*),KPB(*)
     DATA ZERO/0.0/DO/
C-----
C----- INITIALIZE FILE MS
     IF(NSEQ.EQ.1) CALL INITMS(NELT,NSEM,MS)
C----- SAVE UNMODIFIED VECTOR VFG (BY B.C.) IN VECTOR VRES
     DO 10 I=1,NEQ
10    VRES(NDLT+I)=VRES(NDLT+I)+VFG(I)
     IF(M.GE.2) WRITE(MP,2000) (VFG(I),I=1,NEQ)
2000  FORMAT(/' GLOBAL LOAD VECTOR NON MODIFIED BY B.C. (FG)' )

```

```

1/(1X,10E12.5))
C----- DETERMINE IF THE GLOBAL MATRIX IS TO BE ASSEMBLED
      IKT=0
      IF(NCLNZ.NE.0) IKT=1
      IF(IASSEL.EQ.1) IKT=1
      IF(IKT.EQ.0) GO TO 50
C----- DETERMINE IF THE GLOBAL MATRIX IS TO BE INITIALIZED
      IF(NSEQ.EQ.1) GO TO 30
      DO 20 I=1,NEQ
20    VKGD(I)=ZERO
C----- FORM TABLES EB AND PB DEFINING EQUATION BLOCKS
30    CALL EQBLOC(KLD,NLBL,NBLM,NEQ,KEB,KPB)
      WRITE(MP,2010) NBLM
2010  FORMAT(15X,'NUMBER OF BLOCKS IN KG (NBLM)=',I5)
      IF(M.LT.2) GO TO 40
      WRITE(MP,2020) (KEB(I),I=1,NBLM+1)
2020  FORMAT(' FIRST EQUATION IN EACH BLOCK (EB)'/(5X,20I5))
      WRITE(MP,2030) (KPB(I),I=1,NBLM)
2030  FORMAT(' FIRST BLOCK CONNECTED TO EACH BLOCK: (PB)'/(5X,20I5))
C----- ASSEMBLE KG, MODIFY FG FOR THE B.C.
40    CALL ASKGD(KLD,VDIMP,KLOCE,VCORE,KCODE,KINTF,VPREG,VPREE,KNE,
      1 VKE,VFE,VKGS,VKGD,VKGI,VFG,KNPEG,KSDEG,VDLE,VDLG,KEB)
      IF(ISEQ.NE.0) GO TO 50
C----- PRINT FG
      IF(M.GE.2) WRITE(MP,2040) (VFG(I),I=1,NEQ)
2040  FORMAT(' GLOBAL LOAD VECTOR MODIFIED BY THE B.C. (FG)'
      1 /(1X,10E12.5))
C----- SOLVE
50    CALL SOLD(VKGS,VKGD,VKGI,VFG,KLD,NEQO,MP,IKT,1,NSYM,ENERG,KEB,
      1 KPB,M4,M7,NLBL,NBLM)
      IF(ISEQ.NE.0) GO TO 60
      IF(NSYM.NE.1) WRITE(MP,2050) ENERG
2050  FORMAT(15X,'ENERGY (ENERG)=',1E12.5)
      IF(M.LT.2) GO TO 60
C----- PIVOTS OF KG AND DETERMINANT
      CALL PRPVTS(VKGD)
C----- UPDATE THE SOLUTION
60    DO 70 I=1,NEQ
70    VDLG(I)=VDLG(I)+VFG(I)
C----- PRINT THE SOLUTION
      WRITE(MP,2060)
2060  FORMAT('// SOLUTION'//)
      CALL PRSOL(KDLNC,VCORG,VDIMP,KNEQ,VDLG)
C----- EVALUATE, UPDATE AND PRINT STRESSES AND STRAINS
      CALL ASGRAD(KLD,VDIMP,KLOCE,VCORE,KCODE,KINTF,VPREG,VPREE,KNE,
      1 VKE,VFE,VKGS,VKGD,VKGI,VFG,KNPEG,KSDEG,VDLE,VDLG,NRES)
      IF(ISEQ.NE.1) GO TO 90
      DO 80 I=1,NEQ
80    VDLG(I)=ZERO
C----- FOR THE PROBLEM WITHOUT CONSTRUCTION SEQUENCES
C----- EVALUATE AND PRINT EQUILIBLIUM RESIDUAL VECTOR
C----- ASSEMBLE THE RESIDUALS
90    IF(ISEQ.NE.0) RETURN
      DO 100 I=1,NEQ

```

```

100  VRES(I)=VRES(NDLT+I)
     CALL ASRESD(1,1,KLD,VDIMP,KLOCE,VCORE,VPREG,VPREE,KNE,VKE,VFE,
1  VKGS,VKGD,VKGI,VFG,KNPEG,KSDEG,VDLE,VDLG,VRES,VRES(NEQ+1))
C----- PRINT THE RESIDUALS
     WRITE(MP,2070)
2070 FORMAT('' EQUILIBRIUM RESIDUALS AND REACTIONS'')
     CALL PRSOL(KDLNC,VCORG,VRES(NEQ+1),KNEQ,VRES)
     RETURN
    END
    SUBROUTINE EQBLOC(KLD,NLBL,NBLMAX,NEQ,KEB,KPB)
C=====
C      TO FORM TABLES KEB AND KPB DEFINING EQUATION BLOCKS
C      INPUT
C          KLD      ARRAY OF ADDRESS OF COLUMN TOP TERMS IN KG
C          NLBL     BLOCKS LENGTH
C          NBLMAX   MAX. NUMBER OF BLOCKS ALLOWED
C          NEQ      NUMBER OF EQUATIONS
C      OUTPUT
C          KEB      ARRAY CONTAINING THE NUMBERS OF FIRST EQUATIONS IN
C                     EACH BLOCK (DIMENSION NEQ+1)
C          KPB      ARRAY CONTAINING THE NUMBER OF FIRST BLOCKS CONNECTED
C                     TO EACH BLOCK (DIMENSION NEQ)
C          NBLMAX   NUMBER OF BLOCKS
C=====
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
DIMENSION KLD(*),KEB(*),KPB(*)
C-----
C----- FIRST BLOCK
     ILBL=0
     NBL=1
     KEB(1)=1
     KPB(1)=1
     IMIN=1
C----- FOR EACH EQUATION
     DO 70 IK=1,NEQ
C----- ADDRESSES FOR COLUMN IK
     JHK=KLD(IK)
     JHK1=KLD(IK+1)
     LBK1=JHK1-JHK
     IF(LBK1.LE.NLBL) GO TO 10
     WRITE(MP,2000) IK,LBK1,NLBL
2000 FORMAT(' *** ERROR,COLUMN',I5,' GREATER(',I5,') THAN BLOCK(' ,I5,'
1 ')
     STOP
C----- CHECK FOR NEW BLOCK
10    ILBL=ILBL+LBK1
     IF(ILBL.LE.NLBL) GO TO 60
     NBL=NBL+1
     IF(NBL.LE.NBLMAX) GO TO 20
     WRITE(MP,2010) IK
2010 FORMAT(' *** ERROR, EXCESSIVE NUMBER OF BLOCKS, EQUATION',I5)
     STOP
20    KEB(NBL)=IK
     ILBL=LBK1

```

```

C----- SEARCH FOR FIRST BLOCK CONNECTED TO COMPLETED BLOCK
IB=NBL
40 IF(IMIN.GE.KEB(IB)) GO TO 50
IB=IB-1
GO TO 40
50 KPB(NBL-1)=IB
IMIN=IK
C----- SEARCH FOR MINIMUM ROW NUMBER FOR COLUMN TOP TERMS
60 I=IK-LBK1+1
IF(I.LT.IMIN) IMIN=I
70 CONTINUE
C----- FIRST BLOCK CONNECTED TO LAST BLOCK
IB=NBL
80 IF(IMIN.GE.KEB(IB)) GO TO 90
IB=IB-1
GO TO 80
90 KPB(NBL)=IB
KEB(NBL+1)=NEQ+1
NBLMAX=NBL
RETURN
END
SUBROUTINE ASKGD(KLD,VDIMP,KLOCE,VCORE,KCODE,KINTF,VPREG,VPREE,
1 KNE,VKE,VFE,VKGS,VKGD,VKGI,VFG,KNPEG,KSDEG,VDLE,VDLG,KEB)
C=====
C TO ASSEMBLE GLOBAL MATRIX KG (ELEMENT FUNCTION TYPE 3)
C TAKING INTO ACCOUNT OF PRESCRIBED NON ZERO D.O.F.
C VERSION : MATRIX KG STORED BLOCKWISE ON FILE M4
C=====
IMPLICIT REAL*8(A-H,O-Z)
COMMON/COND/NCLT,NCLZ,NCLNZ
COMMON/PREL/NGPE,NPRE,NPRM,NMNL,NHIS
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1 NPEM,NELS
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
COMMON/RGDT/IEL,IKEL,ITPE,ITPE1,ISDE,IMODEL,IFEL,IDLE,ICE,INEL,
1 IPG,ICODE,IMATD,INSE,INPE,IDL0,INEL0,IPGO,IRES
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
COMMON/SEQU/NSEQ,NSEQ0,ISEQ,IASSEL,IBAR
COMMON/NLIN/EPSDL,OMEGA,DPAS,NPAS,NITER,ITER,IMETH,ILAG,INLN
COMMON/DISK/NLBL,NBLM
DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCORE(*),KCODE(*),VPREG(*),
1 VPREE(*),KNE(*),VKE(*),VFE(*),VKGS(*),VKGD(*),VKGI(*),VFG(*),
1 KNPEG(*),KSDEG(*),VDLE(*),VDLG(*),KEB(*)
DATA ZERO/0.D0/
C-----
C----- REWIND FILE M4
REWIND M4
C----- LOOP OVER THE BLOCKS
DO 80 IB=1,NBLM
C----- INITIALIZE THE BLOCK
DO 10 I=1,NLBL
IF(NSYM.EQ.1) VKGI(I)=ZERO
10 VKGS(I)=ZERO
IE1=KEB(IB)

```

```

IE2=KEB(IB+1)-1
C----- LOOP OVER THE ELEMENTS
DO 70 IEL=1,NELT
  IGPE=KNPEGIEL)
C----- READ AN ELEMENT ON FILE ME
  CALL RDELEM(ME,KLOCE,VCORE,KNE)
  IC=(IGPE-1)*(NPRE+NPRM)
  DO 15 I=1,NPRE+NPRM
  15 VPREE(I)=VPREG(IC+I)
C----- GENERATE ELEMENT D.O.F. FOR LAGRAGIAN FORMULATION
  IF(ILAG.GT.0) CALL DLELM(KLOCE,VDLG,VDIMP,VDLE)
  IMODEL=KCODE(IGPE)
  IFEL=KINTF(IGPE)
  IF(NSDG.GT.0) ISDE=KSDEGIEL)
C----- CKECK IF BLOCK IS AFFECTED BY THIS ELEMENT
  DO 20 ID=1,IDL
    J=KLOCE(ID)
    IF(J.LT.IE1.OR.J.GT.IE2) GO TO 20
    GO TO 40
  20 CONTINUE
  30 IF(IB.NE.1.OR.(NCLNZ.EQ.0.AND.IB.EQ.1)) GO TO 70
C----- EVALUATE INTERPOLATION FUNCTIONS IF REQUIRED
  40 IF(ITPE.EQ.ITPE1) GO TO 50
    ICODE=2
    CALL ELEMELB(VCORE,VPREE,VDLE,VKE,VFE)
C----- FORM ELEMENT MATRIX
  50 ICODE=3
    CALL ELEMELB(VCORE,VPREE,VDLE,VKE,VFE)
C----- PRINT ELEMENT MATRIX
    IF(M.LT.2) GO TO 60
    IF(NSYM.EQ.0) IKE=IDL*(IDL+1)/2
    IF(NSYM.EQ.1) IKE=IDL*IDL
    WRITE(MP,2000) IEL,(VKE(I),I=1,IKE)
  2000 FORMAT(' MATRIX ( KE ) , ELEMENT:',I5/(10X,10E12.5))
C----- MODIFY FG FOR THE PRESCRIBED NON ZERO D.O.F.
  60 IF(NCLNZ.NE.0.AND.IB.EQ.1) CALL MODFG(IDL,NSYM,KLOCE,VDIMP,VKE,
     1 VFG)
C----- ASSEMBLE
    CALL ASSELD(1,0,IDL,NSYM,IE1,IE2,KLOCE,KLD,VKE,VFE,VKGS,VKGD,
    1 VKGI,VFG)
    ITPE1=ITPE
  70 CONTINUE
C----- END OF A BLOCK
    WRITE(M4) (VKGS(I),I=1,NLBL)
    IF(NSYM.EQ.1) WRITE(M4) (VKGI(I),I=1,NLBL)
    IF(M.LT.2) GO TO 80
    WRITE(MP,2010) IB,(VKGS(I),I=1,NLBL)
  2010 FORMAT(' UPPER TRIANGLE BLOCK OF ( KG ) NO:',I5/(1X,10E12.5))
    IF(NSYM.EQ.1) WRITE(MP,2020) IB,(VKGI(I),I=1,NLBL)
  2020 FORMAT(' LOWER TRIANGLE BLOCK OF ( KG ) NO:',I5/(1X,10E12.5))
  80 CONTINUE
    IF(M.GE.2) WRITE(MP,2030) (VKGD(I),I=1,NEQ)
  2030 FORMAT(' DIAGONAL OF ( KG )'/(1X,10E12.5))
    RETURN

```

```

END
SUBROUTINE ASSELD(IKG,IFG,IDL,NSYM,IE1,IE2,KLOCE,KLD,VKE,VFE,
1 VKGS,VKGD,VKGI,VFG)
C=====
C TO ASSEMBLE ELEMENT MATRIX (SYMMETRIC OR NOT) AND/OR VRCTOR.
C THE MATRIX IS STORED BLOCKWISE ON DISK
C INPUT
C     IKG      IF IKG.EQ.1 ASSEMBLE ELEMENT MATRIX KE
C     IFG      IF IFG.EQ.1 ASSEMBLE ELEMENT VECTOR FE
C     IDL      NUMBER OF D.O.F. OF THE ELEMENT
C     NSYM    0=SYMMETRIC PROBLEM, 1=NON SYMMETRIC PROBLEM
C     IE1,IE2 FIRST AND LAST COLUMN OF KG TO BE ASSEMBLED
C     KLOCE   ELEMENT LOCALIZATION VECTOR
C     KLD     CUMULATIVE COLUMN HEIGHTS IN KG
C     VKE     ELEMENT MATRIX KE (FULL OR UPPER TRIANGLE BY
C             DESCENDING COLUMNS)
C     VFE     ELEMENT VECTOR FE
C OUTPUT
C     VKGS,VKGD,VKGI   GLOBAL MATRIX (SKYLINE)
C             (SYMMETRIC OR NOT)
C     VFG     GLOBAL LOAD VECTOR
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION KLOCE(*),KLD(*),VKE(*),VFE(*),VKGS(*),VKGD(*),
1 VKGI(*),VFG(*)
C-----
C----- ASSEMBLE ELEMENT MATRIX
IF(IKG.NE.1) GO TO 100
IOBLOC=KLD(IE1)-1
IEQO=IDL
IEQ1=1
C----- FOR EACH COLUMN OF KE
DO 90 JD=1,IDL
IF(NSYM.NE.1) IEQO=JD
JL=KLOCE(JD)
IF(JL) 90,90,10
10 IO=KLD(JL+1)-IOBLOC
IEQ=IEQ1
IQ=1
IF(JL.LT.IE1.OR.JL.GT.IE2) GO TO 90
C----- FOR EACH LOW OF KE
DO 80 ID=1,IDL
IL=KLOCE(ID)
IF(NSYM.EQ.1) GO TO 30
IF(ID-JD) 30,20,20
20 IQ=ID
30 IF(IL) 80,80,40
40 IJ=JL-IL
IF(IJ) 80,50,60
C----- DIAGONAL TERMS IN KG
50 VKGD(IL)=VKGD(IL)+VKE(IEQ)
GO TO 80
C----- UPPER TRIANGLE TERMS IN KG
60 I=IO-IJ

```

```

VKGS(I)=VKGS(I)+VKE(IEQ)
IF(NSYM.NE.1) GO TO 80
C----- LOWER TRIANGLE TERMS IN KG
IEQI=(ID-1)*IDLE+JD
VKGI(I)=VKGI(I)+VKE(IEQI)
80 IEQ=IEQ+IQ
90 IEQI=IEQI+IEQ0
C----- ASSEMBLE ELEMENT VECTOR
100 IF(IFG.NE.1) GO TO 130
DO 120 ID=1, IDLE
IL=KLOCE(ID)
IF(IL) 120,120,110
110 VFG(IL)=VFG(IL)+VFE(ID)
120 CONTINUE
130 RETURN
END
SUBROUTINE SOLD(VKGS,VKGD,VKGI,VFG,KLD,NEQ,MP,IFAC,ISOL,NSYM,ENERG
1 ,KEB,KPB,M4,M7,NLBL,NBLM)
C=====
C TO SOLVE A LINEAR SYSTEM (SYMMETRICAL OR NOT).
C THE MATRIX IS STORED ON FILE M4 BY SKYLINES.
C AFTER TRIANGULARIZATION IT IS STORED ON FILE M5
C INPUT .
C VKGS,VKGD,VKGI      SYSTEM MATRIX : UPPER, DIAGONAL AND LOWER
C                           PARTS
C VFG                  SECOND MEMBER
C KLD                  ADDRESSES OF COLUMN TOP TERMS
C NEQ                  NUMBER OF EQUATIONS
C MP                   OUTPUT DEVICE NUMBER
C IFAC                 IF IFAC.EQ.1 TRIANGULARIZATION OF
C                           THE MATRIX
C ISOL                 IF ISOL.EQ.1 COMPUTE SOLUTION FROM THE
C                           TRIANGULARIZED MATRIX
C NSYM                 INDEX FOR NON SUMMETRIC PROBLEM
C KEB                  NUMBER OF FIRST EQUATION IN EACH BLOCK
C KPB                  NUMBER OF FIRST BLOCK CONNECTED TO EACH
C                           BLOCK
C OUTPUT
C VKGS,VKGD,VKGI      TRIANGULARIZED MATRIX (IF IFAC.EQ.1)
C VFG                  SOLUTION (IF ISOL.EQ.1)
C ENERG                SYSTEM ENERGY (IF NSYM.EQ.0)
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION VKGS(*),VKGD(*),VKGI(*),VFG(*),KLD(*),KEB(*),KPB(*)
DATA ZERO/0.0D/
C-----
REWIND M4
REWIND M7
IK=1
IF(VKGD(1).NE.ZERO) GO TO 5
WRITE(MP,2000) IK
STOP
5 ENERG=ZERO
C----- FOR EACH BLOCK TO BE TRIANGULARIZED

```

```

J1MIN=NLBL+1
J1MAX=NLBL+NLBL
DO 105 IB=1,NLBL
C----- READ A BLOCK TO BE TRIANGULARIZED
READ(M4) (VKGS(I),I=1,NLBL)
IF(NSYM.EQ.1) READ(M4) (VKGI(I),I=1,NLBL)
C----- PARAMETERS FOR BLOCK IB
IKO=KEB(IB)
IK1=KEB(IB+1)-1
IB0=KPB(IB)
JO=KLD(IKO)-1
IF(IB0.EQ.IB) GO TO 11
C----- BACKSPACE ON CONNECTED BLOCKS
I1=IB-IB0
DO 10 I=1,I1
BACKSPACE M7
IF(NSYM.EQ.1) BACKSPACE M7
10 CONTINUE
C----- FOR EACH CONNECTED BLOCK (INCLUDING BLOCK IB ITSELF)
11 DO 103 IBC=IB0,IB
IF(IBC.EQ.IB) GO TO 12
READ(M7) (VKGS(I),I=J1MIN,J1MAX)
IF(NSYM.EQ.1) READ(M7) (VKGI(I),I=J1MIN,J1MAX)
C----- PARAMETERS OF CONNECTED BLOCK
12 IIO=KEB(IBC)
II1=KEB(IBC+1)-1
JCO=KLD(IIO)-1
IF(IBC.NE.IB) JCO=JCO-NLBL
C----- FOR EACH COLUMN OF BLOCK IB TO BE MODIFIED
DO 100 IK=IK0,IK1
JHK=KLD(IK)-JO
C----- ADDRESS OF NEXT COLUMN TOP TERM IK+1
JHK1=KLD(IK+1)-JO
C----- HEIGHT OF COLUMN IK (INCLUDE UPPER AND DIAGONAL TERMS)
LHK=JHK1-JHK
LHK1=LHK-1
C----- ROW OF FIRST TERM TO BE MODIFIED IN COLUMN IK
IMIN=IK-LHK1
IMIN1=IMIN-1
C----- ROW OF LAST TERM TO BE MODIFIED IN COLUMN IK
IMAX=IK-1
IF(LHK1.LT.0) GO TO 100
IF(IFAC.NE.1) GO TO 90
IF(NSYM.EQ.0) GO TO 14
IB1=IB
IF(IMIN1.LT.IKO) IB1=IB0
IF(IBC.EQ.IB1) VKGI(JHK)=VKGI(JHK)/VKGD(IMIN1)
14 IF(IBC.EQ.IB.AND.IK.EQ.IKO) GO TO 40
IF(LHK1.EQ.0) GO TO 40
C----- FIND FIRST AND LAST ROW OF COLUMN IK AFFECTED
C     BY CONNECTED BLOCK IBC
IMINC=MAX0(IMIN,IIO)
IMAXC=MIN0(IMAX,II1)
IF(IMINC.GT.IMAXC) GO TO 40

```

```

C----- MODIFY NON DIAGONAL TERMS OF COLUMN IK
JCK=JHK+IMINC-IMIN1
JHJ=KLD(IMINC)-JCO
C----- FOR EACH TERM TO BE MODIFIED, LOCATED AT JCK
DO 30 IJ=IMINC,IMAXC
JHJ1=KLD(IJ+1)-JCO
C----- NUMBER OF MODIFICATIVE TERMS OF COEFFICIENT LOCATED AT JCK
IC=MINO(JCK-JHK,JHJ1-JHJ)
IF(IC.LE.0.AND.NSYM.EQ.0) GO TO 20
C1=ZERO
IF(IC.LE.0) GO TO 17
J1=JHJ1-IC
J2=JCK-IC
IF(NSYM.EQ.1) GO TO 15
VKGS(JCK)=VKGS(JCK)-SCAL(VKGS(J1),VKGS(J2),IC)
GO TO 20
15 VKGS(JCK)=VKGS(JCK)-SCAL(VKGI(J1),VKGI(J2),IC)
C1=SCAL(VKGS(J1),VKGI(J2),IC)
17 VKGI(JCK)=(VKGI(JCK)-C1)/VKGD(IJ)
20 JCK=JCK+1
30 JHJ=JHJ1
C----- MODIFY DIAGONAL TERM
40 IF(IBC.NE.IB) GO TO 90
JCK=JHK
CDIAG=ZERO
DO 70 IJ=IMIN1,IMAX
C1=VKGS(JCK)
IF(NSYM.EQ.1) GO TO 50
C2=C1/VKGD(IJ)
VKGS(JCK)=C2
GO TO 60
50 C2=VKGI(JCK)
60 CDIAG=CDIAG+C1*C2
70 JCK=JCK+1
VKGD(IK)=VKGD(IK)-CDIAG
IF(VKGD(IK)) 90,80,90
80 WRITE(MP,2000) IK
2000 FORMAT(' *** ERROR, ZERO PIVOT EQUATION ',I5)
STOP
C----- SOLVE LOWER TRIANGULAR SYSTEM
90 IF(ISOL.NE.1) GO TO 100
IF(IBC.NE.IB) GO TO 100
IF(NSYM.NE.1) VFG(IK)=VFG(IK)-SCAL(VKGS(JHK),VFG(IMIN1),LHK)
IF(NSYM.EQ.1) VFG(IK)=VFG(IK)-SCAL(VKGI(JHK),VFG(IMIN1),LHK)
100 CONTINUE
C----- NEXT CONNECTED BLOCK
103 CONTINUE
C----- END OF ELIMINATION OF THIS BLOCK
IF(IB.EQ.NBLM) GO TO 105
WRITE(M7) (VKGS(I),I=1,NLBL)
IF(NSYM.EQ.1) WRITE(M7) (VKGI(I),I=1,NLBL)
105 CONTINUE
IF(ISOL.NE.1) RETURN
C----- SOLVE DIAGONAL SYSTEM

```

```

IF(NSYM.EQ.1) GO TO 120
DO 110 IK=1,NEQ
C1=VKGD(IK)
C2=VFG(IK)/C1
VFG(IK)=C2
110 ENERG=ENERG+C1*C2*C2
C----- SOLVE UPPER TRIANGULAR SYSTEM
120 IB=NBLM
IKO=KEB(IB)-1
JO=KLD(IKO+1)-1
IK=NEQ+1
JHK1=KLD(IK)-JO
C----- FOR EVERY EQUATION FROM NEQ TO 1
130 IK=IK-1
C----- READ A BLOCK IF REQUIRED
IF(IK.NE.IKO) GO TO 135
BACKSPACE M7
IF(NSYM.EQ.1) BACKSPACE M7
READ(M7) (VKGS(I),I=1,NLBL)
IF(NSYM.EQ.1) READ(M7) (VKGI(I),I=1,NLBL)
BACKSPACE M7
IF(NSYM.EQ.1) BACKSPACE M7
IB=IB-1
IKO=KEB(IB)-1
JO=KLD(IKO+1)-1
JHK1=KLD(IK+1)-JO
C----- MODIFY THE UNKNOWN VECTOR
135 IF(NSYM.EQ.1) VFG(IK)=VFG(IK)/VKGD(IK)
IF(IK.EQ.1) RETURN
C1=VFG(IK)
JHK=KLD(IK)-JO
JBK=JHK1-1
IF(JHK.GT.JBK) GO TO 150
IJ=IK-JBK+JHK-1
DO 140 JCK=JHK,JBK
VFG(IJ)=VFG(IJ)-VKGS(JCK)*C1
140 IJ=IJ+1
150 JHK1=JHK
GO TO 130
END
SUBROUTINE BLNLND
=====
C TO CALL BLOCK 'NLND'
C TO SOLVE A STEADY NON LINEAR PROBLEM
=====
IMPLICIT REAL*8(A-H,O-Z)
CHARACTER*4 TBL
COMMON/COOR/NDIM,NNT,NDLN,NDLT
COMMON/PREL/NGPE,NPRE,NPRM,NMNL,NHIS
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,LAG,NIDENT,NPG,NSEM,
1 NPEM,NELS
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
COMMON/NLIN/EPSDL,OMEGA,DPAS,NPAS,NITER,ITER,IMETH,ILAG,INLN
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH

```

```

COMMON/ALLO/NVA,IVA,IVAMAX,NREEL,NLBL
COMMON/LOCA/LCORG,LDLNC,LNEQ,LDIMP,LCODE,LINTF,LPREG,LLD,LLOCE,
1 LCORE,LNE,LPREE,LNPEG,LSDEG,LDLE,LKE,LFE,LKGS,LKGD,LKG,I,LFG,
2 LRES,LDLG,LENELS,LEB,LPB
COMMON/SEQU/NSEQ,NSEQO,ISEQ,IASSEL,IBAR
COMMON/DISK/NLBL,NBLM
COMMON VA(20000)
DIMENSION TBL(11)
DATA TBL/'KGS ','KGD ','KGI ','FG ','KE ','FE ','RES ','DLE ',
1 'DLG ','LEB ','LPB '/,DEUX/2.00/,NBLMAX/100/
C-----
C----- ILAG=NLAG
C----- INLN=1
C----- NSEQ=NSEQ+1
C----- NSEQO=NSEQO+1
C----- WRITE(MP,2000) M
2000 FORMAT('' ON DISK ASSEMBLAGE AND NONLINEAR SOLUTION (M=',
1 I2,')'/1X,50('='))
IF(NSEQO.NE.1) GO TO 10
IF(NMNL.EQ.0) GO TO 10
C----- OPEN DIRECT ACCESS FILE M3 (ELASTO-PLASTIC MATRICES)
LD=16*8
OPEN(M3,FILE='M3.DAT',STATUS='NEW',ACCESS='DIRECT',RECL=LD)
C----- OPEN DIRECT ACCESS FILE M9 (STIFFNESS IDENTIFIER DATA)
LT=NPEM*4
OPEN(M9,FILE='M9.DAT',STATUS='NEW',ACCESS='DIRECT',RECL=LT)
C----- OPEN DIRECT ACCESS FILE MH (PATH HISTORY DATA)
IF(NHIS.EQ.0) GO TO 10
LH=NPEM*NHIS*8
OPEN(MH,FILE='MH.DAT',STATUS='NEW',ACCESS='DIRECT',RECL=LH)
10 IF(NSEQ.NE.1) GO TO 50
C----- OPEN DIRECT ACCESS FILE MS (STRESSES, STRAINS AND PWPS)
LS=NSEM*8
OPEN(MS,FILE='MS.DAT',STATUS='NEW',ACCESS='DIRECT',RECL=LS)
C----- OPEN SEQUENTIAL FILES, M4 AND M7 (STORE MATRIX KG)
OPEN(M4,FILE='M4.DAT',STATUS='NEW',FORM='UNFORMATTED')
OPEN(M7,FILE='M7.DAT',STATUS='NEW',FORM='UNFORMATTED')
C----- TO ALLOCATE SPACE
IF(LKGD.EQ.1) CALL ESPACE(NEQ,1,TBL(2),LKGD)
IF(LFG.EQ.1) CALL ESPACE(NEQ,1,TBL(4),LFG)
IF(LKE.EQ.1) CALL ESPACE(NKE,1,TBL(5),LKE)
IF(LFE.EQ.1) CALL ESPACE(NDLE,1,TBL(6),LFE)
IF(LRES.EQ.1) CALL ESPACE(NDLT+NEQ,1,TBL(7),LRES)
IF(LDLE.EQ.1) CALL ESPACE(NDLE,1,TBL(8),LDLE)
IF(LDLG.EQ.1) CALL ESPACE(NEQ,1,TBL(9),LDLG)
C----- FIND BLOCK LENGTH
NLBL=0
NBLM=0
I3=2
I2=1+NSYM
IF(NLBL.EQ.0) GO TO 20
IF(NBLM.EQ.0) NBLM=NKG/NLBL+2
GO TO 40
20 I1=NVA-IVA-(2*NBLMAX+2)/NREEL-1

```

```

IF(I1.GE.(NKG*I2+2)) GO TO 30
C----- CASE WHERE MATRIX KG IS TO BE SEGMENTED
NLBL=I1/(DEUX*I2)
NBLM=NKG/NLBL+2
C----- CASE WHERE MATRIX KG IS IN CORE
30 NLBL=NKG
NBLM=1
I3=1
40 WRITE(MP,2010) NLBL,NBLM
2010 FORMAT(15X,'BLOCKS LENGTH IN KG' (NLBL)=',I5/
1      15X,'MAX. NUMBER OF BLOCKS IN KG' =',I5)
CALL ESPACE(NBLM+1,0,TBL(10),LEB)
CALL ESPACE(NBLM,0,TBL(11),LPB)
IF(LKGS.EQ.1) CALL ESPACE(NLBL*I3,1,TBL(1),LKGS)
IF(NSYM.EQ.1.AND.LKG1.EQ.1) CALL ESPACE(NLBL*I3,1,TBL(3),LKGI)
C----- TO EXECUTE THE BLOCK
50 CALL EXNLND(VA(LLD),VA(LDIMP),VA(LLOCE),VA(LCORE),VA(LCODE),
1 VA(LINTF),VA(LPREG),VA(LPREE),VA(LNE),VA(LKE),VA(LFE),VA(LKGS),
2 VA(LKGD),VA(LKG1),VA(LFG),VA(LCORG),VA(LDLNC),VA(LNEQ),
3 VA(LNPEG),VA(LRES),VA(LSDEG),VA(LDLE),VA(LDLG),VA(LEB),VA(LPB))
RETURN
END
SUBROUTINE EXNLND(KLD,VDIMP,KLOCE,VCORE,KCODE,KINTF,VPREG,VPREE,
1 KNE,VKE,VFE,VKGS,VKGD,VKG1,VFG,VCORG,KDLNC,KNEQ,KNPEG,VRES,
2 KSDEG,VDLE,VDLG,KEB,KPB)
C=====
C----- TO EXECUTE BLOCK 'NLND'
C----- ASSEMBLE AND SOLVE A NON LINEAR PROBLEM BY NEWTON-RAPHSON METHOD
C----- WHEN MATRIX KG IS STORED BLOCKWISE ON DISK
C----- IMETH.EQ.1 COMPUTE K AT EACH ITERATION
C----- IMETH.EQ.2 K IS CONSTANT AT EACH ITERATION
C----- IMETH.EQ.3 RECOMPUTE K AT THE BEGINNING OF EACH STEP
C----- IMETH.EQ.4 K IS LINEAR AT THE FIRST ITERATION(DEFAULT)
C=====
IMPLICIT REAL*8(A-H,O-Z)
COMMON/COOR/NDIM,NNT,NDLN,NDLT
COMMON/PREL/NGPE,NPRE,NPRM,NMNL,NHIS
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDLE
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1 NPEM,NELS
COMMON/NLIN/EPSDL,OMEGA,DPAS,NPAS,NITER,ITER,IMETH,ILAG,INLN
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
COMMON/SEQU/NSEQ,NSEQ0,ISEQ,IASSEL,IBAR
COMMON/DISK/NLBL,NBLM
DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCORE(*),KCODE(*),KINTF(*),
1 VPREG(*),VPREE(*),KNE(*),VKE(*),VFE(*),VKGS(*),VKGD(*),VKG1(*),
2 VFG(*),VCORG(*),KDLNC(*),KNEQ(*),KNPEG(*),VRES(*),KSDEG(*),
3 VDLE(*),VDLG(*),KEB(*),KPB(*)
DATA ZERO/0.D0/,UN/1.D0/
C-----
C----- INITIALIZE FILE MS
IF(NSEG.EQ.1) CALL INITMS(NELT,NSEM,MS)
IF(NSEQ0.NE.1) GO TO 10
C----- INITIALIZE FILES M9 AND MH IF DEAL WITH NONLINEAR MATERIAL

```

```

IF(NMNL.EQ.1) CALL INITM9(M9,NELT,NPEM)
IF(NHIS.GT.0) CALL INITMH(VPREG,VPREE,KNPEG,KCODE,MS,MH,NPRE,
1 NPRM,NELT)
10 DPAS0=ZERO
XPAS=ZERO
IPAS=0
C----- READ A CARD DEFINING A SET OF IDENTICAL STEPS
READ(MR,1000) DPAS,I1,I2,I3,X1,X2,IELS,IEQL
1000 FORMAT(F10.0,3I5,2F10.0,2I5)
IF(I1.GT.0) NPAS=I1
IF(I2.GT.0) NITER=I2
IF(I3.GT.0) IMETH=I3
IF(X1.GT.ZERO) EPSDL=X1
IF(X2.GT.ZERO) OMEGA=X2
C----- LOOP OVER ALL STEPS
DO 130 IP=1,NPAS
IPAS=IPAS+1
XPAS=XPAS+DPAS
WRITE(MP,2000) IPAS,DPAS,XPAS,NITER,IMETH,EPSDL,OMEGA
2000 FORMAT(/1X,13(' -'),'STEP NUMBER (IPAS):',I5 //
1          14X,'INCREMENT                               (DPAS)=',E12.5/
2          14X,'TOTAL LEVEL                           (XPAS)=',E12.5/
3          14X,'NUMBER OF ITERATIONS                (NITER)=',I12/
4          14X,'METHOD NUMBER                         (IMETH)=',I12/
5          14X,'TOLERANCE                            (EPSDL)=',E12.5/
6          14X,'OVER RELAXATION FACTOR           (OMEGA)=',E12.5)
C----- INCREMENT THE LOADS AND STORE IN THE RESIDUAL VECTOR
CALL MAJ(DPAS,ZERO,NEQ,VFG,VRES)
C----- UPDATE TOTAL LOAD VECTOR
DO 20 I=1,NEQ
20 VRES(NDLT+I)=VRES(NDLT+I)+VRES(I)
C----- LOOP OVER EQUILIBRIUM ITERATIONS
DO 110 ITER=1,NITER
IF(IMETH.GT.4) WRITE(MP,2005) IMETH
2005 FORMAT(' ** ERROR, METHOD:',I3,' UNKNOWN')
IF(IMETH.GT.4) STOP
C----- DECIDE IF GLOBAL MATRIX IS TO BE ASSEMBLED
IKT=0
IF(IMETH.EQ.1.OR.IMETH.EQ.4) IKT=1
IF(IMETH.EQ.2.AND.IASSEL.EQ.1.AND.IPAS.EQ.1.AND.ITER.EQ.1) IKT=1
IF(IMETH.EQ.3.AND.ITER.EQ.1) IKT=1
IF(IKT.EQ.0) GO TO 60
C----- INITIALIZE GLOBAL MATRIX TO ZERO IF IT IS TO BE ASSEMBLED
IF(NSEQ.EQ.1.AND.IPAS.EQ.1.AND.ITER.EQ.1) GO TO 40
DO 30 I=1,NEQ
30 VKGD(I)=ZERO
C----- FORM TABLES EB AND PB DEFINING EQUATION BLOCKS
40 CALL EQBLOC(KLD,NLBL,NBLM,NEQ,KEB,KPB)
WRITE(MP,2010) NBLM
2010 FORMAT(15X,'NUMBER OF BLOCKS IN KG (NBLM)=',I5)
IF(M.LT.2) GO TO 50
WRITE(MP,2020) (KEB(I),I=1,NBLM+1)
2020 FORMAT(/' FIRST EQUATION IN EACH BLOCK (EB)'/(5X,20I5))
WRITE(MP,2030) (KPB(I),I=1,NBLM)

```

```

2030 FORMAT(' FIRST BLOCK CONNECTED TO EACH BLOCK: (PB)'/(5X,20I5))
C----- ASSEMBLE KG
50    CALL ASKGD(KLD,VDIMP,KLOCE,VCORE,KCODE,KINTF,VPREG,VPREE,KNE,
1   VKE,VFE,VKGS,VKGD,VKGI,VFG,KNPEG,KSDEG,VDLE,VDLG,KEB)
C----- SOVLE
60    CALL SOLD(VKGS,VKGD,VKGI,VRES,KLD,NEQ,MP,IKT,1,NSYM,ENERG,KEB,
1   KPB,M4,M7,NLBL,NBLM)
    IF(IKT.EQ.1.AND.M.GT.1) CALL PRPVTS(VKGD)
C----- UPDATE STRESSES, STRAINS AND PWPS
    CALL ASGRAD(KLD,VDIMP,KLOCE,VCORE,KCODE,KINTF,VPREG,VPREE,KNE,
1   VKE,VFE,VKGS,VKGD,VKGI,VRES,KNPEG,KSDEG,VDLE,VDLG,NRES)
C----- UPDATE THE SOLUTION
    CALL MAJ(OMEGA,UN,NEQ,VRES,VDLG)
C----- COMPUTE THE NORM
    CALL NORME(NEQ,VRES,VDLG,XNORM)
    IF(M.GT.0) WRITE(MP,2040) ITER,XNORM
2040 FORMAT(5X,'ITERATION (ITER):',I3,' NORM (XNORM)=',E12.5)
    IF(M.GE.2) CALL PRSOL(KDLNC,VCORG,VDIMP,KNEQ,VDLG)
C----- UPDATE STRESSES, STRAINS AND PWPS
C    CALL ASGRAD(KLD,VDIMP,KLOCE,VCORE,KCODE,VPREG,VPREE,KNE,VKE,
C 1   VFE,VKGS,VKGD,VKGI,VRES,KNPEG,KSDEG,VDLE,VDLG,NRES)
    IF(NRES.EQ.0) GO TO 120
    IF(XNORM.LE.EPSDL) GO TO 120
C----- ASSEMBLE RESIDUAL VECTOR
    DO 70 I=1,NEQ
70    VRES(I)=ZERO
    CALL ASRESD(1,0,KLD,VDIMP,KLOCE,VCORE,VPREG,VPREE,KNE,VKE,VFE,
1   VKGS,VKGD,VKGI,VFG,KNPEG,KSDEG,VDLE,VDLG,VRES,VRES(NEQ+1))
    DO 80 I=1,NEQ
80    VRES(I)=VRES(NDLT+I)-VRES(I)
110  CONTINUE
    ITER=NITER
C----- END OF STEP
120  DPASO=DPAS
    WRITE(MP,2050) ITER,NITER
2050 FORMAT('/10X,I4,' PERFORMED ITERATIONS OVER',I4/')
    IF(M.LT.2) CALL PRSOL(KDLNC,VCORG,VDIMP,KNEQ,VDLG)
    IF(IELS.EQ.1) CALL ELSOL(KLD,VDIMP,KLOCE,VCORE,VPREE,KNE,VKE,
1   VFE,VKGS,VKGD,VKGI,VDLE,VDLG,VRES,IPAS,NLAG,IEQL)
130  CONTINUE
C----- FOR THE PROBLEM WITHOUT CONSTRUCTION SEQUENCES
C----- EVALUATE AND PRINT EQUILIBRIUM RESIDUAL VECTOR
    IF(IEQ.NE.0) RETURN
C----- SAVE VECTOR VFG IN VRES AND CHANGE SIGN
    DO 140 I=1,NEQ
140  VRES(I)=-VRES(NDLT+I)
C----- ASSEMBLE THE RESIDUALS
    CALL ASRESD(1,1,KLD,VDIMP,KLOCE,VCORE,VPREG,VPREE,KNE,VKE,VFE,
1   VKGS,VKGD,VKGI,VFG,KNPEG,KSDEG,VDLE,VDLG,VRES,VRES(NEQ+1))
    WRITE(MP,2060)
2060 FORMAT('/* EQUILIBRIUM RESIDUALS AND REACTIONS'//)
    CALL PRSOL(KDLNC,VCORG,VRES(NEQ+1),KNEQ,VRES)
    RETURN
    END

```

```

SUBROUTINE ELEMLB(VCORE,VPREE,VDLE,VKE,VFE)
C=====
C      TO COMPUTE ELEMENT INFORMATIONS FOR ALL TYPES OF ELEMENTS
C=====
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/RGDTIEL,IKE,ITPE,ITPE1,ISDE,IMODEL,IFEL,IDE,ICE,INEL,
      1  IPG,ICODE,IMATD,INSE,INPE,IDL0,INEL0,IPGO,IRES
      DIMENSION VCORE(*),VPREE(*),VDLE(*),VKE(*),VFE(*)
C-----
      GO TO ( 10, 20, 30, 40, 50, 60, 70, 80, 90, 100),ITPE
C----- ELEMENT OF TYPE 1
10    CALL EL1D(VCORE,VPREE,VDLE,VKE,VFE)
      GO TO 900
C----- ELEMENT OF TYPE 2
20    CALL EL2D(VCORE,VPREE,VDLE,VKE,VFE)
      GO TO 900
C----- ELEMENT OF TYPE 3
30    CALL EL2D(VCORE,VPREE,VDLE,VKE,VFE)
      GO TO 900
C----- ELEMENT OF TYPE 4
40    CONTINUE
      GO TO 900
C----- ELEMENT OF TYPE 5
50    CONTINUE
      GO TO 900
C----- ELEMENT OF TYPE 6
60    CONTINUE
      GO TO 900
C----- ELEMENT OF TYPE 7
70    CONTINUE
      GO TO 900
C----- ELEMENT OF TYPE 8
80    CONTINUE
      GO TO 900
C----- ELEMENT OF TYPE 9
90    CONTINUE
      GO TO 900
C----- ELEMENT OF TYPE 10
100   CONTINUE
      GO TO 900
C----- OTHER ELEMENTS
C      .....
900   RETURN
      END
      SUBROUTINE EL1D(VCORE,VPREE,VDLE,VKE,VFE)
C=====
C      2 NODES LINEAR BAR ELEMENT FOR 1 DIMENSIONAL ELASTICITY
C      EVALUATE ELEMENT INFORMATIONS ACCORDING TO ICODE VALUE
C          ICODE=1 ELEMENT PARAMETERS
C          ICODE=3 EVALUATE ELEMENT STIFFNESS MATRIX
C          ICODE=5 EVALUATE NODAL FORCES AT DELETED BAR NODES
C          ICODE=8 EVALUATE STRESSES AND STRAINS
C          ICODE=9 EVALUATE NODAL FORCES AT BAR NODES
C          ICODE=10 PRINT STRESSES AND STRAINS

```

```

C      ELEMENT PROPERTIES
C      VPREE(1)  YOUNG'S MODULUS
C      VPREE(2)  SECTION AREA
C=====
C      IMPLICIT REAL*8(A-H,O-Z)
COMMON/PREL/NGPE,NPRE,NPRM,NMNL,NHIS
COMMON/RGDTIEL,IKELEM,INSE,INPE,IDL0,INELO,IPGO,IRES
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
COMMON/NLIN/EPSDL,OMEGA,DPAS,NPAS,NITER,ITER,IMETH,ILAG,INLN
DIMENSION VCORE(*),VPREE(*),VDLE(*),VKE(*),VFE(*),VSE(2)
C-----
C      DATA ZERO/0.0D0/
C      IMATD=1
C
C----- CHOOSE FUNCTION TO BE EXECUTED
C
C      GO TO (100,200,300,400,500,600,700,800,900,1000),ICODE
C
C----- RETURN ELEMENT PARAMETERS IN COMMON 'RGDT'
C
100   IDL0=4
      INELO=2
      IPGO=1
      RETURN
200   CONTINUE
      RETURN
C
C----- EVALUATE ELEMENT STIFFNESS MATRIX
C
300   DX=VCORE(3)-VCORE(1)
      DY=VCORE(4)-VCORE(2)
      D=DSQRT(DX**2+DY**2)
      SIN=DY/D
      COS=DX/D
      FAC=VPREE(1)*VPREE(2)/D
      X=ZERO
      IF(ILAG.GT.0) READ(MS,REC=IEL) VSE(1)
      IF(ILAG.GT.0) X=VSE(1)/D
      VKE(1)=FAC*COS*COS+X
      VKE(2)=FAC*COS*SIN
      VKE(3)=FAC*SIN*SIN+X
      VKE(4)=-VKE(1)-X
      VKE(5)=-VKE(2)
      VKE(6)=VKE(1)+X
      VKE(7)=-VKE(2)
      VKE(8)=-VKE(3)-X
      VKE(9)=VKE(2)
      VKE(10)=VKE(3)+X
      RETURN
C
C----- NO BODY FORCE FOR BAR ELEMENT
C
400   DO 410 I=1,IDL0

```

```

410  VFE(I)=ZERO
      RETURN
C
C----- EVALUATE NODAL FORCES AT DELETED BAR NODES
C
500  READ(MS,REC=IEL) VSE(1)
      FORCE=VSE(1)
      DX=VCORE(3)-VCORE(1)
      DY=VCORE(4)-VCORE(2)
      DL=DSQRT(DX**2+DY**2)
      SIN=DY/DL
      COS=DX/DL
      VFE(1)=-FORCE*COS
      VFE(2)=-FORCE*SIN
      VFE(3)=-VFE(1)
      VFE(4)=-VFE(2)
      VSE(1)=ZERO
      VSE(2)=ZERO
      WRITE(MS,REC=IEL) VSE(1),VSE(2)
      RETURN
600  CONTINUE
      RETURN
700  CONTINUE
      RETURN
C
C----- EVALUATE BAR FORCE AND STRAIN
C----- ASSUME PERFECTLY PLASTIC BEHAVIOR FOR NONLINEAR MATERIAL
C
800  IRES=0
      READ(MS,REC=IEL) VSE(1),VSE(2)
      DX=VCORE(3)-VCORE(1)
      DY=VCORE(4)-VCORE(2)
      D=DSQRT(DX**2+DY**2)
      SIN=DY/D
      COS=DX/D
      SWI=VDLE(1)*COS+VDLE(2)*SIN
      SWJ=VDLE(3)*COS+VDLE(4)*SIN
      STRAIN=(SWJ-SWI)/D
      FORCE=STRAIN*VPREE(1)*VPREE(2)
      VSE(1)=VSE(1)+FORCE
      VSE(2)=VSE(2)+STRAIN
      IF(IMODEL.EQ.0) GO TO 810
      YIELD=VPREE(NPRE+1)
      IF(DABS(VSE(1)).LT.YIELD) GO TO 810
      IF(VSE(1).GT.YIELD) VSE(1)=YIELD*VPREE(2)
      IF(VSE(1).LT.-YIELD) VSE(1)=-YIELD*VPREE(2)
      IRES=1
810  WRITE(MS,REC=IEL) VSE(1),VSE(2)
      RETURN
C
C----- EVALUATE THE ELEMENT EQUILIBRIUM RESIDUALS
C
900  READ(MS,REC=IEL) VSE(1)
      DX=VCORE(3)-VCORE(1)

```

```

DY=VCORE(4)-VCORE(2)
D=DSQRT(DX**2+DY**2)
SIN=DY/D
COS=DX/D
FORCE=VSE(1)
VFE(1)=-FORCE*COS
VFE(2)=-FORCE*SIN
VFE(3)=-VFE(1)
VFE(4)=-VFE(2)
RETURN

C
C----- PRINT FORCES AND STRAINS
C
1000 READ(MS,REC=IEL) VSE(1),VSE(2)
FORCE=VSE(1)
STRAIN=VSE(2)
WRITE(MP,2090) IEL,FORCE,STRAIN
2090 FORMAT('/' SOLUTION IN ELEMENT ',I5
1 'BAR FORCE = ',E12.5/'BAR STRAIN = ',E12.5)
RETURN
END
SUBROUTINE EL2D(VCORE,VPREE,VDLE,VKE,VFE)
=====
C      TWO DIMENSIONAL ELASTICITY AND PLASTICITY INFORMATIONS
C      ITPE=2   4 NODES QUADRILATERAL ELEMENT
C      ITPE=3   8 NODES QUADRATIC ELEMENT
C      EVALUATE ELEMENT INFORMATIONS ACCORDING TO ICODE VALUE
C      ICODE=1   ELEMENT PARAMETERS
C      ICODE=2   EVALUATE INTERPOLATION FUNCTIONS
C      ICODE=3   EVALUATE ELEMENT STIFFNESS MATRIX
C      ICODE=4   EQUIVALENT NODAL LOADS DUE TO BODY FORCES
C      ICODE=5   EQUIVALENT NODAL LOADS DUE TO EXCAVATION
C      ICODE=6   ..... NOT WRITTEN ..... (FOR FUTURE USE)
C      ICODE=7   ..... NOT WRITTEN ..... (FOR FUTURE USE)
C      ICODE=8   EVALUATE STRESSES, STRAINS AND PWPS
C      ICODE=9   EQUIVALENT NODAL LOADS DUE TO STRESSES AT G.P.
C      ICODE=10  PRINT STRESSES, STRAINS, PWPS, PRINCIPAL STRESSES
C      ELEMENT PROPERTIES
C      VPREE(1) YOUNG'S MODULUS
C      VPREE(2) POISSON'S COEFFICIENT
C      VPREE(3) UNIT WEIGHT (TOTAL)
C      VPREE(4) COEFFICIENT OF LATERAL EARTH PRESSURE(IF NEEDED)
=====
IMPLICIT REAL*8(A-H,O-Z)
COMMON/COOR/NDIM,NNT,NDLN,NDLT
COMMON/PREL/NGPE,NPRE,NPRM,NMNL,NHIS
COMMON/ELEM/NELT,NNEL,NKEL,NTPE,NSDG,NLAG,NIDENT,NPG,NSEM,
1 NPEM,NELS
COMMON/ASSE/NEQ,NSYM,NKG,NKE,NDEL
COMMON/RGDT/IEL,IKEL,ITPE,ITPE1,ISDE,IMODEL,IFEL,IDL,ICE,INEL,
1 IPG,ICODE,IMATD,INSE,INPE,IDL0,INEL0,IPGO,IRES
COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
COMMON/SEQU/NSEQ,NSEQ0,ISEQ,IASSEL,IBAR
COMMON/NLIN/EPSDL,OMEGA,DPAS,npas,NITER,IMETH,ILAG,INLN

```

```

COMMON/TVL3/YHAT,ROWA,BKWA,ATMP
DIMENSION VCORE(*),VPREE(*),VDLE(*),VKE(*),VFE(*)
DIMENSION SIG(4),EPS(4),SIGO(4),XSTART(9),HIST(9),VSE(81),
1 VCE(25),MT(9)
C----- CHARACTERISTIC DIMENSIONS OF THE ELEMENT
C     DIMENSION VCPG(      IPG),VKPG(NDIM*IPG),VDE1(IMATD**2)
C     DIMENSION VCPG(         9),VKPG(       18),VDE1(       16)
C     DIMENSION VBE(   5*IDLE),VDE(IMATD**2),VJ(NDIM**2),VJ1(NDIM**2)
C     DIMENSION VBE(      80),VDE(      16),VJ(       4),VJ1(       4)
C     DIMENSION VNIX( INEL*NDIM),VNI ((1+NDIM)*INEL*IPG),IPGKED(NDIM)
C     DIMENSION VNIX(      16),VNI (        216),IPGKED(      2)
C----- NUMBER OF G.P.
C     DATA IPGKED/3,3/
C----- DATA ZERO/0.D0/,DEUX/2.D0/,X05/0.5D0/,RADN/.572957795130823D2/
C     DATA EPX/1.D-6/
C     SQRT(X)=DSQRT(X)
C     ATAN2(X,Y)=DATAN2(X,Y)
C     IMATD=4
C     PI=4.D0*DATAN(1.D0)
C----- CHOOSE FUNCTION TO BE EXECUTED
C     GO TO (100,200,300,400,500,600,700,800,900,1000),ICODE
C----- RETURN ELEMENT PARAMETERS IN COMMON 'RGDT'
C
100    IPGO=9
      IF(ITPE.EQ.3) GO TO 110
      IDLE0=8
      INELO=4
      RETURN
110    IDLE0=16
      INELO=8
      RETURN
C----- EVALUATE COORDINATES, WEIGHTS, FUNCTIONS N AND THEIR
C----- DERIVATIVES AT G.P.
C
200    CALL GAUSS(IPGKED,NDIM,VKPG,VCPG,IPG)
      IF(M.LT.2) GO TO 220
      WRITE(MP,2000) IPG
2000  FORMAT(/I5,' GAUSS POINTS'/10X,'VCPG',25X,'VKPG')
      IO=1
      DO 210 IG=1,IPG
      I1=IO+NDIM-1
      WRITE(MP,2010) VCPG(IG),(VKPG(I),I=IO,I1)
210    IO=IO+NDIM
2010  FORMAT(1X,F20.15,5X,3F20.15)
220    CALL NIQ(VKPG,VNI)
      IF(M.LT.2) RETURN
      I1=3*INEL*IPG
      WRITE(MP,2020) (VNI(I),I=1,I1)
2020  FORMAT(/' FUNCTIONS N AND DERIVATIVES'/(1X,8E12.5))

```

```

      RETURN
C
C----- EVALUATE ELEMENT STIFFNESS MATRIX
C
C----- INITIALIZE VKE
300  DO 310 I=1,NKE
310  VKE(I)=ZERO
    IF(SEQ.EQ.1) VPREE(2)=VPREE(4)/(1.0+VPREE(4))
    IF(SEQ.EQ.1.AND.ISDE.EQ.2) ISDE=1
C----- CHECK IF NEED TO READ ELEMENT STRESS DATA
    IMS=0
    IF(INLN.EQ.0) IMODEL=0
    IF(IMETH.EQ.4.AND.ITER.EQ.1) IMODEL=0
    IF(ILAG.GT.0) IMS=1
    IF(IMODEL.GT.0) IMS=1
    IF(IMS.GT.0) READ(MS,REC=IEL) (VSE(I),I=1,INSE)
C----- CHECK IF NEED TO READ STRESS STATE AND HISTORY DATA
    IF(IMODEL.GT.0) READ(M9,REC=IEL) (MT(I),I=1,IPG)
    IF(IMODEL.GT.0.AND.NHIS.GT.0) CALL RDMH(MH,IEL,IPG,IMODEL,
1 XSTART,HIST)
C----- UPDAT COORDINATES FOR UPDATED LAGRANGIAN FORMULATION
    IF(ILAG.GT.0) CALL UPCORE(VCORE,VDLE,IDLE)
C----- FORM ELASTIC STRESS-STRAIN MATRIX D FOR LINEAR MATERIAL
    IF(IMODEL.GT.0) GO TO 320
    CALL D02(VPREE,VDE,IKEL,IFEL)
    IF(ISDE.EQ.2) CALL DU2(VDE,BKWA)
    IF(M.GE.2) WRITE(MP,2030) (VDE(I),I=1,16)
2030 FORMAT(/' MATRIX D'/1X,16E12.5)
C----- LOOP OVER G.P.
320  II=1+INEL
    I2=1
    DO 370 IG=1,IPG
    IF(IMS.EQ.0) GO TO 330
C----- INPUT DATA AT A G.P. FOR NONLINEAR PROBLEM
    SIG(1)=VSE(IG*4-3)
    SIG(2)=VSE(IG*4-2)
    SIG(3)=VSE(IG*4-1)
    SIG(4)=VSE(IG*4)
330  IF(IMODEL.EQ.0) GO TO 340
    MTYPE=MT(IG)
    IF(IMODEL.EQ.4) EL=HIST(IG)
C----- FORM TANGENT STRESS-STRAIN MATRIX D FOR ALL TYPES OF MODELS
    IF(IMODEL.EQ.1) CALL FVM2D(VPREE,SIG,VDE,ATMP,IKEL,NPRE)
    IF(IMODEL.EQ.2) CALL FMC2D(VPREE,SIG,VDE,IKEL,NPRE,MTYPE)
    IF(IMODEL.EQ.3) CALL FDP2D(VPREE,SIG,VDE,IKEL,NPRE,MTYPE)
    IF(IMODEL.EQ.4) CALL FCP2D(VPREE,SIG,VDE,EL,IKEL,NPRE,MTYPE)
    IF(IMODEL.EQ.5) CALL FVS2D(VPREE,VCORE,SIG,VDE,ROWA,ATMP,NPRE)
C----- ADD OTHER MODELS HERE IF NEEDED
    IV=(IEL-1)*IPG+IG
    WRITE(M3,REC=IV) (VDE(I),I=1,16)
    IF(ISDE.EQ.2) CALL DU2(VDE,BKWA)
C----- EVALUATE THE JACOBIAN, ITS INVERSE AND ITS DETERMINANT
340  CALL JACOB(VNI(I1),VCORE,NDIM,INEL,VJ,VJ1,DETJ)
    IF(DETJ.LT.EPX) WRITE(MP,2040) IEL,IG,DETJ

```

```

2040 FORMAT(' *** ELEM ',I5,' G.P. ',I3,' DET(J)=' ,E12.5)
      IF(M.GE.2) WRITE(MP,2050) VJ,VJ1,DETJ
2050 FORMAT(/' JACOBIAN=' ,4E12.5, ' / J INVERS=' ,4E12.5, ' DETJ=' ,E12.5)
C----- PERFORM D*COEF
      C=VCPG(IG)*DETJ
      DO 350 I=1,16
350  VDE1(I)=VDE(I)*C
C----- COMPUTE DERIVATIVES OF INTERPOLATION FUNCTIONS WRT X,Y
      CALL DNIDX(VNI(I1),VJ1,NDIM,INEL,VNIX)
      IF(M.GE.2) WRITE(MP,2060) (VNIX(I),I=1,16)
2060  FORMAT('/ VNIX'/(1X,8E12.5))
C----- FIND RADIUS FOR AXISYMMETRIC CASE
      IF(IKEL.EQ.3) CALL RADIUS(VCORE,VNI(I2),R,INEL)
C----- FORM MATRIX B FOR REGULAR OR UPDATED LAGRANGIAN FORMULATION
      CALL BO2(VNIX,VNI(I2),INEL,IKEL,VBE,R)
      IF(M.GE.2) WRITE(MP,2070) (VBE(I),I=1,64)
2070  FORMAT('/ MATRIX B'/(1X,10E12.5))
C----- FORM LINEAR COMPONENTS IN ELEMENT STIFFNESS MATRIX
      CALL BTDB(VKE,VBE,VDE1,IDL,IMATD,IKEL,NSYM)
      IF(ILAG.EQ.0) GO TO 360
C----- FORM CAUCHY STRESS MATRIX FOR UPDATED LAGRANGIAN FORMULATION
      CALL CO2(SIG,VCE,C)
C----- FORM NONLINEAR STRAIN-DISPLACEMENT TRANSFORMATION MATRIX
      CALL BN2(VNIX,VNI(I2),INEL,IKEL,VBE,R)
C----- ADD NONLINEAR COMPONENTS IN ELEMENT STIFFNESS MATRIX
      CALL BTDB(VKE,VBE,VCE,IDL,5,IKEL,NSYM)
360   I2=I2+3*INEL
370   I1=I1+3*INEL
      RETURN
C
C----- EVALUATE EQUIVALENT NODAL FORCES DUE TO BODY FORCES
C----- FX,FY PER UNIT VOLUME (FOR GRAVITY FX=0, FY=-VPREE(3))
C
400  FX=ZERO
      FY=-VPREE(3)
      IF(ISEQ.GT.0.AND.ISEQ.EQ.1) FY=ROWA-VPREE(3).
      IF(ISEQ.EQ.2) FY=-ROWA
      IF(ISEQ.EQ.8) FY=ROWA
      IF(ISEQ.EQ.2.OR.ISEQ.EQ.8) CALL WATTBL(VCORE,YWAT,IDL,ISEQ)
      DO 410 I=1,IDL
410  VFE(I)=ZERO
      I1=1
      DO 430 IG=1,IPG
      CALL JACOB(VNI(I1+INEL),VCORE,NDIM,INEL,VJ,VJ1,DETJ)
      IF(IKEL.EQ.3) CALL RADIUS(VCORE,VNI(I1),R,INEL)
      DX=VCPG(IG)*DETJ
      DY=DX*FY
      DX=DX*FX
      IF(IKEL.EQ.3) DY=DY*2.D0*PI*R
      I2=I1
      I3=1
      DO 420 IN=1,INEL
      VFE(I3)=VFE(I3)+DX*VNI(I2)
      VFE(I3+1)=VFE(I3+1)+DY*VNI(I2)

```

```

        I2=I2+1
420    I3=I3+2
430    I1=I1+3*INEL
C----- INITIALIZE ELEMENT HISTORY DATA FOR THE ADDED ELEMENT
        IF(ISEQ.NE.4) RETURN
        IF(IMODEL.EQ.0) RETURN
        DO 440 I=1,IPG
          MT(I)=1
440    IF(IMODEL.EQ.4) HIST(I)=VPREE(NPRE+8)
        WRITE(M9,REC=IEL) (MT(I),I=1,IPG)
        CALL WRMH(MH,IEL,IPG,IMODEL,XSTART,HIST)
        RETURN

C
C----- EVALUATE EQUIVALENT NODAL FORCES ON EXCAVATED SURFACE
C
500    NS=INSE*2+INPE
        READ(MS,REC=IEL) (VSE(I),I=1,INSE)
        FY=VPREE(3)
        DO 510 I=1,IDLE
510    VFE(I)=ZERO
          I1=1
C----- ELEMENT NODAL LOADS DUE TO BODY FORCE
        DO 530 IG=1,IPG
        CALL JACOB(VNI(I1+INEL),VCORE,NDIM,INEL,VJ,VJ1,DETJ)
        IF(IKEL.EQ.3) CALL RADIUS(VCORE,VNI(I1),R,INEL)
        DX=VCPG(IG)*DETJ
        DY=DX*FY
        IF(IKEL.EQ.3) DY=DY*2.00*PI*R
        I2=I1
        I3=1
        DO 520 IN=1,INEL
          VFE(I3+1)=VFE(I3+1)+DY*VNI(I2)
          I2=I2+1
520    I3=I3+2
530    I1=I1+3*INEL
C----- ELEMENT NODAL FORCES DUE TO STRESSES AT G.P.
        IF(ILAG.EQ.1) CALL UPCORE(VCORE,VDLE,IDL)
        I1=1
        DO 550 IG=1,IPG
        CALL JACOB(VNI(I1+INEL),VCORE,NDIM,INEL,VJ,VJ1,DETJ)
        CALL DNIDX(VNI(I1+INEL),VJ1,NDIM,INEL,VNIX)
        IF(IKEL.EQ.3) CALL RADIUS(VCORE,VNI(I1),R,INEL)
        DX=VCPG(IG)*DETJ
        IF(IKEL.EQ.3) DX=DX*2.00*PI*R
        SIG(1)=VSE(IG*4-3)*DX
        SIG(2)=VSE(IG*4-2)*DX
        SIG(3)=VSE(IG*4-1)*DX
        IF(IKEL.EQ.3) SIG(4)=VSE(IG*4)*DX
        I2=I1
        I3=1
        DO 540 IN=1,INEL
          C1=VNIX(IN)
          IN1=IN+INEL
          C2=VNIX(IN1)

```

```

VFE(I3)=VFE(I3)+C1*SIG(1)+C2*SIG(3)
VFE(I3+1)=VFE(I3+1)+C2*SIG(2)+C1*SIG(3)
IF(IKEL.NE.3) GO TO 540
VFE(I3)=VFE(I3)+VNI(I2)*SIG(4)/R
I2=I2+1
540 I3=I3+2
550 I1=I1+3*INEL
C----- SET VECTOR VSE TO ZERO FOR THE EXCAVATED ELEMENT
      DO 560 I=1,NS
560 VSE(I)=ZERO
      WRITE(MS,REC=IEL) (VSE(I),I=1,NS)
      RETURN
C
C----- NOT WRITTEN
C
600 CONTINUE
      RETURN
C----- NOT WRITTEN
700 CONTINUE
      RETURN
C
C----- EVALUATE STRESSES, STRAINS AND PWPS AT G.P.
C
C----- READ STRESSES, STRAINS AND PWPS IN AN ELEMENT
800 NS=INSE*2+INPE
      READ(MS,REC=IEL) (VSE(I),I=1,NS)
      IF(ISEQ.EQ.1) VPREE(2)=VPREE(4)/(1.0D+VPREE(4))
      IF(ISEQ.EQ.1.AND.ISDE.EQ.2) ISDE=1
      IF(INLN.EQ.0) IMODEL=0
C----- READ STRESS STATE AND HISTORY DATA FOR NONLINEAR MATERIAL
      IF(IMODEL.GT.0) READ(M9,REC=IEL) (MT(I),I=1,IPG)
      IF(IMODEL.GT.0.AND.NHIS.GT.0) CALL RDMH(MH,IEL,IPG,IMODEL,
      1 XSTART,HIST)
C----- UPDATE COORDINATES FOR UPDATED LAGRANGIAN FORMULATION
      IF(ILAG.EQ.1) CALL UPCORE(VCORE,VFE,IDLE)
C----- FORM ELASTIC STRESS-STRAIN MATRIX FOR LINEAR MATERIAL
      IF(IMODEL.EQ.0) CALL D02(VPREE,VDE,IKEL,IFEL)
C----- LOOP OVER THE G.P.
      IRES=0
      I1=1+INEL
      I2=0
      DO 890 IG=1,IPG
C----- READ TANGENT STRESS-STRAIN MATRIX D FROM FILE M3
      IF(IMODEL.EQ.0) GO TO 810
      IF(IMETH.EQ.4.AND.ITER.EQ.1) GO TO 810
      IV=(IEL-1)*IPG+IG
      READ(M3,REC=IV) (VDE(I),I=1,16)
C----- EVALUATE THE JACOBIAN
810 CALL JACOB(VNI(I1),VCORE,NDIM,INEL,VJ,VJ1,DETJ)
C----- EVALUATE FUNCTIONS D(NI)/D(X)
      CALL DNIDX(VNI(I1),VJ1,NDIM,INEL,VNIX)
C----- CALCULATE RADIUS FOR AXISYMMETRIC CASE
      IF(IKEL.EQ.3) CALL RADIUS(VCORE,VNI(I2+1),R,INEL)
C----- PREVIOUS STRESSES

```

```

SIG(1)=VSE(IG*4-3)
SIG(2)=VSE(IG*4-2)
SIG(3)=VSE(IG*4-1)
SIG(4)=VSE(IG*4)
C----- INITIALIZE STRAINS
U11=ZERO
U22=ZERO
U21=ZERO
U12=ZERO
U33=ZERO
C----- COMPUTE LINEAR (INCREMENTAL) STRAINS AT G. P.
ID=1
DO 820 IN=1,INEL
UN=VDLE(ID)
VN=VDLE(ID+1)
C1=VNIX(IN)
IN1=IN+INEL
C2=VNIX(IN1)
U11=U11+C1*UN
U22=U22+C2*VN
U21=U21+C2*UN
U12=U12+C1*VN
IF(IKEL.EQ.3) U33=U33+UN/R
820 ID=ID+2
EPS(1)=U11
EPS(2)=U22
EPS(3)=U21+U12
EPS(4)=U33
IF(ILAG.EQ.0) GO TO 830
C----- ADD SECOND ORDER STRAINS FOR LAGRANGIAN FORMUATION
EPS(1)=EPS(1)+0.5D0*(U11*U11+U21*U21)
EPS(2)=EPS(2)+0.5D0*(U12*U12+U22*U22)
EPS(3)=EPS(3)+U11*U12+U21*U22
IF(IKEL.EQ.3) EPS(4)=EPS(4)+0.5D0*U33*U33
C----- TIME-INDEPENDENT RIGID BODY ROTATION AND STRESS CHANGES
W=0.5D0*(U21-U12)
SW11=2.D0*SIG(3)*W
SW12=(SIG(1)-SIG(2))*W
C----- COMPUTE STRESSES BY ASSUMING (LINEAR) ELASTIC BEHAVIOR
830 CALL MULT(VDE,EPS,SIGO,4,4,1)
SIGO(1)=SIGO(1)+SIG(1)
SIGO(2)=SIGO(2)+SIG(2)
SIGO(3)=SIGO(3)+SIG(3)
SIGO(4)=SIGO(4)+SIG(4)
IF(IMODEL.EQ.0) GO TO 840
C----- INPUT HISTORY DATA AT A G.P. IF NEEDED
IF(IMODEL.EQ.4) ASTART=XSTART(IG)
IF(IMODEL.EQ.4) EL=HIST(IG)
C----- USE SPECIFIED SOIL MODEL TO FIND ACTUAL STRESSES
IF(IMODEL.EQ.1) CALL VARMOD(VPREE,SIG,EPS,ATMP,NPRE,MTYPE)
IF(IMODEL.EQ.2) CALL MC2D(VPREE,SIGO,SIG,EPS,NPRE,MTYPE)
IF(IMODEL.EQ.3) CALL DP2D(VPREE,SIG,EPS,NPRE,MTYPE)
IF(IMODEL.EQ.4) CALL CAP2D(VPREE,SIGO,SIG,EPS,ASTART,EL,NPRE,
1 MTYPE)

```

```

      IF(IMODEL.EQ.5) CALL VSHEAR(VPREE,VCORE,SIG,EPS,ROWA,ATMP,NPRE,
      1 MTYPE)
C----- ADD MORE SOIL MODELS HERE IF NEEDED
C----- UPDATE STRESS STATE AND HISTORY DATA AT A G.P.
      IF(IMODEL.EQ.4) HIST(IG)=EL
      MT(IG)=MTYPE
      IF(MTYPE.LE.1) GO TO 850
      IRES=1
      GO TO 850
C----- STRESSES FOR LINEAR ELASTIC ELEMENT
840   SIG(1)=SIG0(1)
      SIG(2)=SIG0(2)
      SIG(3)=SIG0(3)
      SIG(4)=SIG0(4)
C----- FIND CAUCHY STRESSES FOR UPDATED LAGRANGIAN FORMULATION
850   IF(ILAG.EQ.0) GO TO 860
      SIG(1)=SIG(1)+SW11
      SIG(2)=SIG(2)+SW12
      SIG(3)=SIG(3)+SW11
C----- UPDATE STRESSES AND STRAINS
860   DO 870 I=1,4
      IS=(IG-1)*4+I
      IE=IS+INSE
      VSE(IS)=SIG(I)
870   VSE(IE)=VSE(IE)+EPS(I)
      IF(NSDG,EQ.0) GO TO 880
C----- UPDATE PORE WATER PRESSURES
      IF(ISDE.GT.0) CALL PWPS(VCORE,VNI(I2+1),EPS,BKWA,ROWA,YWAT,
      1 PWP,ISDE,INEL)
      IF(ISDE.EQ.0) VSE(IG+INSE*2)=ZERO
      IF(ISDE.EQ.1) VSE(IG+INSE*2)=PWP
      IF(ISDE.EQ.2) VSE(IG+INSE*2)=VSE(IG+INSE*2)+PWP
880   I2=I2+3*INEL
890   I1=I1+3*INEL
C----- STORE UPDATED STRESSES, STRAINS AND PWPS
      WRITE(MS,REC=IEL) (VSE(I),I=1,NS)
C----- STORE UPDATED ELEMENT HISTORY DATA
      IF(IMODEL.EQ.0) RETURN
      IF(IMODEL.GT.0.AND.NHIS.GT.0) CALL WRMH(MH,IEL,IPG,IMODEL,
      1 XSTART,HIST)
C----- UPDATE STRESS STATE DATA
      WRITE(M9,REC=IEL) (MT(I),I=1,IPG)
      RETURN
C
C----- EVALUATE EQUIVALENT NODAL LOADS DUE TO STRESSES AT G.P.
C
C----- INITIALIZE THE ELEMENT LOAD VECTOR
900   DO 910 ID=1,IDL
910   VFE(ID)=ZERO
C----- READ ELEMENT STRESS DATA
      READ(MS,REC=IEL) (VSE(I),I=1,INSE)
C----- UPDATE COORDINATES FOR UPDATED LAGRANGIAN FORMULATION
      IF(ILAG.EQ.1) CALL UPCORE(VCORE,VDLE,IDL)
C----- LOOP OVER THE G.P.

```

```

I1=1+INEL
I2=0
DO 990 IG=1,IPG
C----- EVALUATE THE JACOBIAN
  CALL JACOB(VNI(I1),VCORE,NDIM,INEL,VJ,VJ1,DETJ)
C----- EVALUATE FUNCTIONS D(NI)/D(X)
  CALL DNIDX(VNI(I1),VJ1,NDIM,INEL,VNIX)
C----- FIND RADIUS AT A G.P. FOR AXISYMMETRIC CASE
  IF(IKEL.EQ.3) CALL RADIUS(VCORE,VNI(I2+1),R,INEL)
C----- INPUT STRESSES
  C1=VCPG(IG)*DETJ
  IF(IKEL.EQ.3) C1=C1*2.D0*PI*R
  IF(ISDE.EQ.2) GO TO 940
  SIG(1)=VSE(IG*4-3)*C1
  SIG(2)=VSE(IG*4-2)*C1
  SIG(3)=VSE(IG*4-1)*C1
  IF(IKEL.EQ.3) SIG(4)=VSE(IG*4)*C1
  GO TO 950
940  SIG(1)=(VSE(IG*4-3)+VSE(INSE*2+IG))*C1
  SIG(2)=(VSE(IG*4-2)+VSE(INSE*2+IG))*C1
  SIG(3)=VSE(IG*4-1)*C1
  IF(IKEL.EQ.3) SIG(4)=(VSE(IG*4)+VSE(INSE*2+IG))*C1
C----- CALCULATE EQUIVALENT NODAL LOADS
C----- FOR REGULAR OR UPDATED LAGRANGIAN FORMULATION
950  ID=1
  DO 960 IN=1,INEL
    C1=VNIX(IN)
    IN1=IN+INEL
    C2=VNIX(IN1)
    VFE(ID)=VFE(ID)+C1*SIG(1)+C2*SIG(3)
    VFE(ID+1)=VFE(ID+1)+C2*SIG(2)+C1*SIG(3)
    IF(IKEL.NE.3) GO TO 960
    IN1=IN+I2
    VFE(ID)=VFE(ID)+VNI(IN1)*SIG(4)/R
960  ID=ID+2
  I2=I2+3*INEL
990  I1=I1+3*INEL
  RETURN
C
C----- PRINT STRESSES, STRAINS, PWPS AND PRINCIPAL STRESSES AT G.P.
C
1000 NS=INSE*2+INPE
  READ(MS,REC=IEL) (VSE(I),I=1,NS)
  WRITE(MP,2080) IEL
2080 FORMAT(//' SOLUTION IN ELEMENT ',I5)
  IF(M.EQ.0) WRITE(MP,2081)
  IF(M.EQ.1) WRITE(MP,2082)
  IF(M.GT.1) WRITE(MP,2083)
2081 FORMAT(
 1' P.G.',7X,'X',11X,'Y',9X,'SIGX',8X,'SIGY',7X,'TAUXY',8X,'PWPS'/)
2082 FORMAT(
 1' P.G.',7X,'X',11X,'Y',9X,'SIGX',8X,'SIGY',7X,'TAUXY',8X,'PWPS'/
 2 35X,'EPSX',8X,'EPSY',7X,'GAUXY'/)
2083 FORMAT(

```

```

1' P.G.',7X,'X',11X,'Y',9X,'SIGX',8X,'SIGY',7X,'TAUXY',8X,'PWPS'/
2 35X,'EPSX',8X,'EPSY',7X,'GAUXY'/35X,'SIG1',8X,'SIG2',7X,'TAUMAX',
3 7X,'TETA')/)

C----- LOOP OVER THE G.P.
I2=0
PWP=ZERO
DO 1040 IG=1,IPG
C----- COMPUTE COORDINATES AT G.P.
X=ZERO
Y=ZERO
ID=1
DO 1010 IN=1,INEL
XN=VCORE(ID)
YN=VCORE(ID+1)
IN1=IN+I2
C3=VNI(IN1)
X=X+C3*XN
Y=Y+C3*YN
1010 ID=ID+2
C----- INPUT STRESSES, STRAINS AND PWP
DO 1020 I=1,3
IS=(IG-1)*4+I
IE=IS+INSE
SIG(I)=VSE(IS)
1020 EPS(I)=VSE(IE)
IF(ISDE.GT.0) PWP=VSE(IG+INSE*2)
IF(M.GT.1) GO TO 1030
IF(M.EQ.1) GO TO 1025
WRITE(MP,2084) IG,X,Y,SIG(1),SIG(2),SIG(3),PWP
2084 FORMAT(1X,I5,6E12.5)
GO TO 1040
1025 WRITE(MP,2085) IG,X,Y,SIG(1),SIG(2),SIG(3),PWP,EPS(1),EPS(2),
1 EPS(3)
2085 FORMAT(1X,I5,6E12.5/30X,3E12.5)
GO TO 1040
C----- COMPUTE THE PRINCIPAL STRESSES
1030 TETA=ATAN2(DEUX*SIG(3),SIG(1)-SIG(2))*X05
TETA=TETA*RADN
C1=(SIG(1)+SIG(2))*X05
C2=(SIG(1)-SIG(2))*X05
TAUMAX=SQRT(C2*C2+SIG(3)*SIG(3))
SIG1=C1+TAUMAX
SIG2=C1-TAUMAX
WRITE(MP,2090) IG,X,Y,SIG(1),SIG(2),SIG(3),PWP,EPS(1),EPS(2),
1 EPS(3),SIG1,SIG2,TAUMAX,TETA
2090 FORMAT(1X,I5,6E12.5/30X,3E12.5/30X,4E12.5)
1040 I2=I2+3*INEL
RETURN
END
SUBROUTINE RDMH(MH,IEL,IPG,IMODEL,XSTART,HIST)
C=====
C READ HISTORY DATA FOR NONLINEAR MATERIAL
C=====
IMPLICIT REAL*8(A-H,O-Z)

```

```

DIMENSION XSTART(*),HIST(*)
IF(IMODEL.EQ.0) RETURN
IF(IMODEL.EQ.1) RETURN
IF(IMODEL.EQ.2) RETURN
IF(IMODEL.EQ.3) RETURN
IF(IMODEL.EQ.4) READ(MH,REC=IEL) (XSTART(I),I=1,IPG),
1 (HIST(I),I=1,IPG)
IF(IMODEL.EQ.5) RETURN
RETURN
END
SUBROUTINE WRMH(MH,IEL,IPG,IMODEL,XSTART,HIST)
C=====
C   WRITE HISTORY DATA FOR NONLINEAR MATERIAL
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION XSTART(*),HIST(*)
IF(IMODEL.EQ.0) RETURN
IF(IMODEL.EQ.1) RETURN
IF(IMODEL.EQ.2) RETURN
IF(IMODEL.EQ.3) RETURN
IF(IMODEL.EQ.4) WRITE(MH,REC=IEL) (XSTART(I),I=1,IPG),
1 (HIST(I),I=1,IPG)
IF(IMODEL.EQ.5) RETURN
RETURN
END
SUBROUTINE UPCORE(VCORE,VDLE,IDLE)
C=====
C   CALCULATE UPDATED COORDINATES FOR UPDATED LAGRANGIAN FORMULATION
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION VCORE(*),VDLE(*)
DO 10 I=1,IDLE
10 VCORE(I)=VCORE(I)+VDLE(I)
RETURN
END
SUBROUTINE RADIUS(VCORE,VNI,R,INEL)
C=====
C   CALCULATE THE RADIUS AT A G.P. FOR THE AXISYMMETRIC CASE
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION VCORE(*),VNI(*)
R=0.D0
ID=1
DO 10 I=1,INEL
X=VCORE(ID)
R=R+VNI(I)*X
10 ID=ID+2
RETURN
END
SUBROUTINE WATLBL(VCORE,YWAT,IDL,ESEQ)
C=====
C   FIND THE NEW WATER TABLE LEVEL
C=====
IMPLICIT REAL*8(A-H,O-Z)

```

```

DIMENSION VCORE(*)
C-----  

C      IF(ISEQ.EQ.8) GO TO 20  

C      DO 10 I=2, IDLE, 2  

10     IF(VCORE(I).LT.YWAT) YWAT=VCORE(I)  

      RETURN  

20     DO 30 I=2, IDLE, 2  

30     IF(VCORE(I).GT.YWAT) YWAT=VCORE(I)  

      RETURN  

      END  

      SUBROUTINE PWPS(VCORE,VNI,EPS,BKWA,ROWA,YWAT,PWP,ISDE,INEL)
C=====  

C      TO FIND PORE WATER PRESSURE FOR SATURATED DRAINED OR  

C      SATURATED UNDRAINED CASE  

C=====  

IMPLICIT REAL*8(A-H,O-Z)
DIMENSION VCORE(*),VNI(*),EPS(*)
C-----  

IF(ISDE.EQ.2) GO TO 20
C---- FIND Y-COORDINATE
Y=0.D0
ID=2
DO 10 I=1, INEL
YN=VCORE(ID)
Y=Y+VNI(I)*YN
10 ID=ID+2
C---- SATURATED DRAINED CASE
PWP=(Y-YWAT)*ROWA
RETURN
C---- SATURATED UNDRAINED CASE
20 PWP=(EPS(1)+EPS(2)+EPS(4))*BKWA
RETURN
END
SUBROUTINE GAUSS(IPGKED,NDIM,VKPG,VCPG,IPG)
C=====  

C      TO FORM ARRAYS OF COORDINATES AND WEIGHTS AT GAUSS POINTS
C      (1,2 AND 3 DIMENSIONS)(1,2,3 OR 4 G.P. PER DIMENSION)
C      INPUT
C          IPGKED    NUMBER OF POINTS IN KSI,ETA,ZETA DIRECTIONS
C          NDIM      NUMBER OF DIMENSIONS (1,2 OR 3)
C      OUTPUT
C          VKPG      COORDINATES OF GAUSS POINTS
C          VCPG      WEIGHTS AT GAUSS POINTS
C          IPG       TOTAL NUMBER OF GAUSS POINTS
C=====  

IMPLICIT REAL*8(A-H,O-Z)
DIMENSION IPGKED(*),VKPG(*),VCPG(*),G(10),P(10),INDIC(4)
DATA INDIC/1,2,4,7/
DATA G/0.0D0,-.577350269189626D0,.577250269189626D0,
1      -.774596669241483D0,0.0D0,.774596669241483D0,
2      -.861136311594050D0,-.339981043584860D0,
3      .339981043584860D0,.861136311594050D0/
DATA P/2.0D0,1.0D0,1.0D0,
1      0.555555555555556D0,0.88888888888889D0,0.55555555555556D0,

```

```

2      .347854845137450D0,.652145154862550D0,
3      .652145154862550D0,.347854845137450D0/
C-----
10      II=IPGKED(1)
11      IMIN=INDIC(II)
12      IMAX=IMIN+II-1
13      IF(NDIM>2) 10,20,30
C----- 1 DIMENSION
14      IPG=0
15      DO 15 I=IMIN,IMAX
16          IPG=IPG+1
17          VKPG(IPG)=G(I)
18      VCPG(IPG)=P(I)
19      RETURN
C----- 2 DIMENSIONS
20      II=IPGKED(2)
21      JMIN=INDIC(II)
22      JMAX=JMIN+II-1
23      IPG=0
24      L=1
25      DO 25 I=IMIN,IMAX
26          DO 25 J=JMIN,JMAX
27              IPG=IPG+1
28              VKPG(L)=G(I)
29              VKPG(L+1)=G(J)
30              L=L+2
31      VCPG(IPG)=P(I)*P(J)
32      RETURN
C----- 3 DIMENSIONS
33      II=IPGKED(2)
34      JMIN=INDIC(II)
35      JMAX=JMIN+II-1
36      II=IPGKED(3)
37      KMIN=INDIC(II)
38      KMAX=KMIN+II-1
39      IPG=0
40      L=1
41      DO 35 I=IMIN,IMAX
42          DO 35 J=JMIN,JMAX
43          DO 35 K=KMIN,KMAX
44              IPG=IPG+1
45              VKPG(L)=G(I)
46              VKPG(L+1)=G(J)
47              VKPG(L+2)=G(K)
48              L=L+3
49      VCPG(IPG)=P(I)*P(J)*P(K)
50      RETURN
51      END
52      SUBROUTINE D02(VPREE,VDE,IKEL,IFEL)
C=====
C      TO FORM STRESS-STRAIN MATRIX D (2 DIMENSIONAL ELASTICITY)
C          IKEL      .EQ.1  PLANE STRESS
C          .EQ.2  PLANE STRAIN
C          .EQ.3  AXISYMMETRIC

```

```

C      IFEL      .EQ.1  PLANE STRAIN (INTERFACE ELEMENT)
C      VDE      MATRIX D (FULL)
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION VPREE(*),VDE(4,4)
DATA ZERO/0.D0/,UN/1.D0/,DEUX/2.D0/
E=VPREE(1)
V=VPREE(2)
IF(IKEL.EQ.1) A=ZERO
IF(IKEL.GT.1) A=UN
C1=E*(UN-A*V)/((UN+V)*(UN-V-A*V))
C2=C1*V/(UN-A*V)
C3=E/(DEUX*(UN+V))
VDE(1,1)=C1
VDE(1,2)=C2
VDE(1,3)=ZERO
VDE(2,1)=C2
VDE(2,2)=C1
VDE(2,3)=ZERO
VDE(3,1)=ZERO
VDE(3,2)=ZERO
VDE(3,3)=C3
IF(IFEL.EQ.1) VDE(3,3)=VPREE(5)
IF(IKEL.EQ.1) C2=ZERO
IF(IKEL.EQ.1) C1=ZERO
VDE(1,4)=C2
VDE(2,4)=C2
VDE(3,4)=ZERO
VDE(4,1)=C2
VDE(4,2)=C2
VDE(4,3)=ZERO
VDE(4,4)=C1
RETURN
END
SUBROUTINE DU2(VDE,BKWA)
C=====
C      TO FIND STRESS-STRAIN MATRIX D FOR UNDRAINED ELEMENT
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION VDE(4,4)
VDE(1,1)=VDE(1,1)+BKWA
VDE(1,2)=VDE(1,2)+BKWA
VDE(1,4)=VDE(1,4)+BKWA
VDE(2,1)=VDE(2,1)+BKWA
VDE(2,2)=VDE(2,2)+BKWA
VDE(2,4)=VDE(2,4)+BKWA
VDE(4,1)=VDE(4,1)+BKWA
VDE(4,2)=VDE(4,2)+BKWA
VDE(4,4)=VDE(4,4)+BKWA
RETURN
END
SUBROUTINE JACOB(VNI,VCORE,NDIM,INEL,VJ,VJ1,DETJ)
C=====
C      TO EVALUATE THE JACOBIAN MATRIX, ITS DETERMINANT AND

```

```

C      ITS INVERSE (1,2,3 DIMENSIONS)
C      INPUT
C          VNI      DERIVATIVES OF INTERPOLATION FUNCTION W.R.T.
C                      KSI,ETA,DZETA
C          VCORE    ELEMENT NODAL COORDINATES
C          NDIM     NUMBER OF DIMENSIONS
C          INEL     NUMBER OF NODES PER ELEMENT
C      OUTPUT
C          VJ       JACOBIAN MATRIX
C          VJ1     INVERSE OF JACOBIAN MATRIX
C          DETJ    DETERMINANT OF JACOBIAN MATRIX
C=====
C      IMPLICIT REAL*8(A-H,O-Z) .
C      DIMENSION VNI(INEL,*),VCORE(NDIM,*),VJ(*),VJ1(*)
C      DATA ZERO/0.D0/,UN/1.D0/
C-----
C----- FORM THE JACOBIAN MATRIX
      J=1
      DO 20 JJ=1,NDIM
      DO 20 II=1,NDIM
      C=ZERO
      DO 10 IJ=1,INEL
10      C=C+VNI(IJ,II)*VCORE(JJ,IJ)
      VJ(J)=C
      20      J=J+1
C----- 1, 2, OR 3 DIMENSIONAL INVERSION
      GO TO (40,50,60),NDIM
40      DETJ=VJ(1)
      IF(DETJ.EQ.ZERO) RETURN
      VJ1(1)=UN/DETJ
      RETURN
50      DETJ=VJ(1)*VJ(4)-VJ(2)*VJ(3)
      IF(DETJ.EQ.ZERO) RETURN
      VJ1(1)=VJ(4)/DETJ
      VJ1(2)=-VJ(2)/DETJ
      VJ1(3)=-VJ(3)/DETJ
      VJ1(4)=VJ(1)/DETJ
      RETURN
60      DETJ=VJ(1)*(VJ(5)*VJ(9)-VJ(8)*VJ(6))
1      +VJ(4)*(VJ(8)*VJ(3)-VJ(2)*VJ(9))
2      +VJ(7)*(VJ(2)*VJ(6)-VJ(5)*VJ(3))
      IF(DETJ.EQ.ZERO) RETURN
      VJ1(1)=(VJ(5)*VJ(9)-VJ(6)*VJ(8))/DETJ
      VJ1(2)=(VJ(3)*VJ(8)-VJ(2)*VJ(9))/DETJ
      VJ1(3)=(VJ(2)*VJ(6)-VJ(3)*VJ(5))/DETJ
      VJ1(4)=(VJ(7)*VJ(6)-VJ(4)*VJ(9))/DETJ
      VJ1(5)=(VJ(1)*VJ(9)-VJ(7)*VJ(3))/DETJ
      VJ1(6)=(VJ(4)*VJ(3)-VJ(6)*VJ(1))/DETJ
      VJ1(7)=(VJ(4)*VJ(8)-VJ(7)*VJ(5))/DETJ
      VJ1(8)=(VJ(2)*VJ(7)-VJ(8)*VJ(1))/DETJ
      VJ1(9)=(VJ(1)*VJ(5)-VJ(4)*VJ(2))/DETJ
      RETURN
      END
      SUBROUTINE DNIDX(VNI,VJ1,NDIM,INEL,VNIX)

```

```

C=====
C      COMPUTE THE DERIVATIVES OF INTERPOLATION FUNCTIONS WITH
C      RESPECT TO X,Y,Z
C      (1,2 OR 3 DIMENSIONS)
C      INPUT
C          VNI      DERIVATIVES OF INTERPOLATION FUNCTIONS WITH RESPECT
C                      TO KSI,ETA,DZETA
C          VJ1      INVERSE OF THE JACOBIAN
C          NDIM     NUMBER OF DIMENSIONS (1,2 OR 3)
C          INEL     NUMBER OF INTERPOLATION FUNCTIONS (OR NODES)
C      OUTPUT
C          VNIX     X,Y,Z DERIVATIVES OF INTERPOLATION FUNCTIONS
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION VNI(INEL,*),VJ1(NDIM,*),VNIX(INEL,*)
DATA ZERO/0.D0/
C-----
      DO 20 I=1,NDIM
      DO 20 J=1,INEL
      C=ZERO
      DO 10 IJ=1,NDIM
10    C=C+VJ1(I,IJ)*VNI(J,IJ)
20    VNIX(J,I)=C
      RETURN
      END
      SUBROUTINE B02(VNIX,VNI,INEL,IKEL,VBE,R)
C=====
C      TO FORM LINEAR STRAIN-DISPLACEMENT MATRIX B (2-D ELASTICITY)
C      FOR REGULAR OR UPDATED LAGRANGIAN FORMULATION
C      INPUT
C          VNIX     DERIVATIVES OF INTERPOLATION FUNCTIONS W.R.T. X,Y,Z
C          INEL     NUMBER OF INTERPOLATION FUNCTIONS
C          OUTPUT
C          VBE     MATRIX B
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION VNIX(INEL,*),VNI(*),VBE(4,*)
DATA ZERO/0.D0/
      J=1
      DO 10 I=1,INEL
      C1=VNIX(I,1)
      C2=VNIX(I,2)
      VBE(1,J)=C1
      VBE(1,J+1)=ZERO
      VBE(2,J)=ZERO
      VBE(2,J+1)=C2
      VBE(3,J)=C2
      VBE(3,J+1)=C1
      IF(IKEL.NE.3) GO TO 10
      VBE(4,J)=VNI(I)/R
      VBE(4,J+1)=ZERO
10    J=J+2
      RETURN
      END

```

```

SUBROUTINE CO2(SIG,VCE,C)
C=====
C      FORM CAUCHY STRESS MATRIX C FOR UPDATED LAGRANGIAN FORMULATION
C      INPUT      SIG  STRESSES AT G.P.
C      C          COEFFICIENT
C      OUTPUT     VCE  MATRIX C
C=====

IMPLICIT REAL*8(A-H,O-Z)
DIMENSION SIG(4),VCE(5,5)
VCE(1,1)=SIG(1)*C
VCE(1,2)=SIG(3)*C
VCE(1,3)=0.D0
VCE(1,4)=0.D0
VCE(1,5)=0.D0
VCE(2,1)=SIG(3)*C
VCE(2,2)=SIG(2)*C
VCE(2,3)=0.D0
VCE(2,4)=0.D0
VCE(2,5)=0.D0
VCE(3,1)=0.D0
VCE(3,2)=0.D0
VCE(3,3)=SIG(1)*C
VCE(3,4)=SIG(3)*C
VCE(3,5)=0.D0
VCE(4,1)=0.D0
VCE(4,2)=0.D0
VCE(4,3)=SIG(3)*C
VCE(4,4)=SIG(2)*C
VCE(4,5)=0.D0
VCE(5,1)=0.D0
VCE(5,2)=0.D0
VCE(5,3)=0.D0
VCE(5,4)=0.D0
VCE(5,5)=SIG(4)*C
RETURN
END

SUBROUTINE BN2(VNIX,VNI,INEL,IKEL,VBE,R)
C=====
C      TO FORM NONLINEAR STRAIN-DISPLACEMENT TRANSFORMATION MATRIX
C      FOR UPDATED LAGRANGIAN FORMULATION
C=====

IMPLICIT REAL*8(A-H,O-Z)
DIMENSION VNIX(INEL,*),VNI(*),VBE(5,*)
DATA ZERO/0.D0/
J=1
DO 10 I=1,INEL
C1=VNIX(I,1)
C2=VNIX(I,2)
VBE(1,J)=C1
VBE(1,J+1)=ZERO
VBE(2,J)=C2
VBE(2,J+1)=ZERO
VBE(3,J)=ZERO
VBE(3,J+1)=C1
10 CONTINUE

```

```

VBE(4,J)=ZERO
VBE(4,J+1)=C2
IF(IKEL.NE.3) GO TO 10
VBE(5,J)=VNI(I)/R
VBE(5,J+1)=ZERO
10   J=J+2
      RETURN
      END
      SUBROUTINE BTDB(VKE,VBE,VDE,IDLE,IMATD,IKEL,NSYM)
C=====
C      TO ADD THE PRODUCT B(T).D.B  TO VKE
C      INPUT
C          VKE      ELEMENT MATRIX NON SYMMETRICAL (NSYM.EQ.1)
C                      SYMMETRICAL (NSYM.EQ.0)
C          VBE      MATRIX B OR NONLINEAR MATRIX B (LAGRANGIAN)
C          VDE      MATRIX D OR CAUCHY STRESS MATRIX C (FULL)
C          IDLE     TOTAL NUMBER OF D.O.F. PER ELEMENT
C          IMATD    DIMENSION OF MATRIX D (MAX. 6)
C      OUTPUT
C          VKE
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION VKE(*),VBE(IMATD,*),VDE(IMATD,*),T(6)
DATA ZERO/0.0D0/
C-----
IM=IMATD
IF(IKEL.LT.3) IM=IM-1
IJ=1
IMAX=IDLE
DO 40 J=1, IDLE
DO 20 I1=1,IM
C=ZERO
DO 10 J1=1,IM
10   C=C+VDE(I1,J1)*VBE(J1,J)
20   T(I1)=C
      IF(NSYM.EQ.0) IMAX=J
      DO 40 I=1,IMAX
      C=ZERO
      DO 30 J1=1,IM
30   C=C+VBE(J1,I)*T(J1)
      VKE(IJ)=VKE(IJ)+C
40   IJ=IJ+1
      RETURN
      END
      SUBROUTINE NIQ(VKPG,VNI)
C=====
C      TO EVALUATE INTERPOLATION FUNCTIONS N AND THEIR DERIVATIVES
C      D(N)/D(KSI) D(N)/D(ETA) BY EXPLICIT FORMULATION
C      (QUADRILATERAL ELEMENTS WITH 4 OR 8 NODES)
C      INPUT
C          VKPG     COORDINATES OF INTEGRATION POINTS
C          IPG      NUMBER OF INTEGRATION POINTS
C      OUTPUT
C          VNI      FUNCTIONS N AND THEIR DERIVATIVES

```

```

C=====
C      IMPLICIT REAL*8(A-H,O-Z)
C      COMMON/COOR/NDIM,NNT,NDLN,NDLT
C      COMMON/RGDTIEL,IKEL,ITPE,ITPE1,ISDE,IMODEL,IFEL,IDL
E,ICE,INEL,
1     IPG,ICODE,IMATD,INSE,INPE,IDELO,INEL0,IPGO,IRES
C      COMMON/NUMB/M,M1,ME,M3,M4,MR,MP,M7,MS,M9,MH
C      DIMENSION VKPG(*),VNI(*)
C      DATA P25/.25D0/,P5/.5D0/,UN/1.D0/,DE/2.D0/
C-----
C----- LOOP OVER INTEGRATION POINTS
II=0
I1=0
DO 10 IG=1,IPG
XG=VKPG(I1+1)
YG=VKPG(I1+2)
C----- 4 NODES ELEMENTS
IF (INEL.NE.4) GO TO 20
C----- FUNCTION N
VNI(II+1)=P25*(UN-XG)*(UN-YG)
VNI(II+2)=P25*(UN+XG)*(UN-YG)
VNI(II+3)=P25*(UN+XG)*(UN+YG)
VNI(II+4)=P25*(UN-XG)*(UN+YG)
C----- KSI DERIVATIVES
VNI(II+5)=-P25*(UN-YG)
VNI(II+6)=P25*(UN-YG)
VNI(II+7)=P25*(UN+YG)
VNI(II+8)=-P25*(UN+YG)
C----- ETA DERIVATIVES
VNI(II+9)=-P25*(UN-XG)
VNI(II+10)=-P25*(UN+XG)
VNI(II+11)=P25*(UN+XG)
VNI(II+12)=P25*(UN-XG)
II=II+12
I1=I1+2
GO TO 10
C----- 8 NODES ELEMENTS
20 IF (INEL.NE.8) GO TO 100
C----- FUNCTION N
VNI(II+1)=P25*(UN-XG)*(UN-YG)*(-XG-YG-UN)
VNI(II+2)=P5*(UN-(XG*XG))*(UN-YG)
VNI(II+3)=P25*(UN+XG)*(UN-YG)*(XG-YG-UN)
VNI(II+4)=P5*(UN+XG)*(UN-(YG*YG))
VNI(II+5)=P25*(UN+XG)*(UN+YG)*(XG+YG-UN)
VNI(II+6)=P5*(UN-(XG*XG))*(UN+YG)
VNI(II+7)=P25*(UN-XG)*(UN+YG)*(YG-XG-UN)
VNI(II+8)=P5*(UN-XG)*(UN-(YG*YG))
C----- KSI DERIVATIVES
II=II+8
VNI(II+1)=P25*((DE*XG)+YG)*(UN-YG)
VNI(II+2)=-XG*(UN-YG)
VNI(II+3)=P25*((DE*XG)-YG)*(UN-YG)
VNI(II+4)=P5*(UN-(YG*YG))
VNI(II+5)=P25*((DE*XG)+YG)*(UN+YG)
VNI(II+6)=-XG*(UN+YG)

```

```

VNI(II+7)=-P25*(YG-(DE*XG))*(UN+YG)
VNI(II+8)=-P5*(UN-(YG*YG))
C----- ETA DERIVATIVES
II=II+8
VNI(II+1)=P25*(UN-XG)*(XG+(DE*YG))
VNI(II+2)=-P5*(UN-(XG*XG))
VNI(II+3)=-P25*(UN+XG)*(XG-(DE*YG))
VNI(II+4)=-YG*(UN+XG)
VNI(II+5)=P25*(UN+XG)*(XG+(DE*YG))
VNI(II+6)=P5*(UN-(XG*XG))
VNI(II+7)=P25*(UN-XG)*((DE*YG)-XG)
VNI(II+8)=-YG*(UN-XG)
II=II+8
I1=I1+2
GO TO 10
100 WRITE(MP,2000)
2000 FORMAT(' ***, ERROR, FUNCTION N NOT DEFINED FOR ELEMENT TYPE')
10 CONTINUE
RETURN
END

```

```

SUBROUTINE VARMOD(VPREE,SIG,EPS,ATMP,NPRE,MTYPE)
C=====
C   VARMOD FINDS THE TANGENT MODULUS AND POISSON RATIO ACCORDING TO
C   A VARIABLE MODULI MODEL BASED ON THE HYPERBOLIC EQUATION
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION VPREE(*),SIG(*),EPS(*)
C----- STATEMENT FUNCTIONS
EI(A,B,C,D)=A*B*(C/B)**D
S(A,B,C,D)=(A-B)*(1.D0-DSIN(C))/(2.D0*(D*DCOS(C)+B*DSIN(C)))
C----- INITIALIZE MODEL PARAMETERS
MTYPE=1
PHIO=VPREE(NPRE+1)
DPHI=VPREE(NPRE+2)
COH=VPREE(NPRE+3)
XK=VPREE(NPRE+4)
EXPN=VPREE(NPRE+5)
RF=VPREE(NPRE+6)
XKB=VPREE(NPRE+7)
EXPM=VPREE(NPRE+8)
XKUR=VPREE(NPRE+9)
C----- INITIALIZE THE TANGENT MODULUS
PI=4.D0*Datan(1.D0)
C----- OBTAIN PRINCIPAL STRESSES. NOTE CHANGE OF SIGN CONVENTIONS
C----- TENSION - NEGATIVE, COMPRESSION - POSITIVE
C1=(-SIG(1)-SIG(2))/2.D0
C2=(SIG(2)-SIG(1))/2.D0
TAUMAX=DSQRT(C2**2+SIG(3)**2)
SIG1=C1+TAUMAX
SIG3=C1-TAUMAX
PHI=PHIO-DPHI*DLOG10(SIG3/ATMP)
PHI=PHI*PI/180.D0
SR=S(SIG1,SIG3,PHI,COH)
ET=EI(XK,ATMP,SIG3,EXPN)*(1.D0-RF*SR)**2
C----- INITIALIZE THE TAGENT POISSON RATIO
BULK=EI(XKB,ATMP,SIG3,EXPM)
VT=.5D0-ET1/BULK/6.D0
IF(VT.LT..2D0) VT=.2D0
C----- INCREMENT THE STRAIN COMPONENTS. CHANGE OF SIGN CONVENTIONS
DEPSX=-EPS(1)/20.D0
DEPSY=-EPS(2)/20.D0
DEPSXY=EPS(3)/20.D0
C----- ITERATE 20 TIMES TO OBTAIN APPROXIMATE ACTUAL STRESSES
DO 100 I=1,20
D11=ET*(1.D0-VT)/((1.D0+VT)*(1.D0-2.D0*VT))
D12=D11*VT/(1.D0-VT)
D33=ET/(2.D0*(1.D0+VT))
SIGX=SIG(1)+D11*DEPSX+D12*DEPSY
SIGY=SIG(2)+D12*DEPSX+D11*DEPSY
SIGXY=SIG(3)+D33*DEPSXY
C1=(-SIGX-SIGY)/2.D0
C2=(SIGY-SIGX)/2.D0
TAUMAX=DSQRT(C2**2+SIGXY**2)
SIG1=C1+TAUMAX

```

```

SIG3=C1-TAUMAX
C----- CHECK IF SOIL IS IN TENSION
IF(SIG1.LT.1.D-20.OR.SIG3.LT.1.D-20) GO TO 30
PHI=PHIO-DPHI*DLOG10(SIG3/ATMP)
PHI=PHI*PI/180.D0
SR=S(SIG1,SIG3,PHI,COH)
C----- COMPUTE THE NEW TANGENT MODULUS
ET=EI(XK,ATMP,SIG3,EXPN)*(1.D0-RF*SR)**2
C----- COMPUTE THE NEW TANGENT POISSON RATIO
BULK=EI(XKB,ATMP,SIG3,EXPX)
VT=.5D0-ET1/BULK/6.D0
IF(VT.LT..2D0) VT=.2D0
MTYPE=2
GO TO 40
C----- NEW STRESSES FOR SOIL IN TENSION
30 ET=VPREE(1)
VT=VPREE(2)
D11=ET*(1.D0-VT)/((1.D0+VT)*(1.D0-2.D0*VT))
D12=D11*VT/(1.D0-VT)
D33=ET/(2.D0*(1.D0+VT))
SIGX=S(SIG1)+D11*DEPSX+D12*DEPSY
SIGY=S(SIG2)+D12*DEPSX+D11*DEPSY
SIGXY=S(SIG3)+D33*DEPSXY
C----- NEW STRESSES AFTER THIS ITERATION
40 SIG(1)=SIGX
SIG(2)=SIGY
SIG(3)=SIGXY
100 CONTINUE
SIG(1)=-SIGX
SIG(2)=-SIGY
SIG(3)=SIGXY
RETURN
END
SUBROUTINE MC2D(VPREE,SIGO,SIG,EPS,NPRE,MTYPE)
C=====
C USE ELASTIC-PERFECTLY PLASTIC MOHR-COULOMB MODEL TO FIND
C ACTUAL STRESSES. STRESSES ARE AXISYMMETRIC. INITIAL STRESS METHOD
C MTYPE=1(LINEAR BEHAVIOR), MTYPE=2(PLASTIC BEHAVIOR)
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION VPREE(*),SIGO(*),SIG(*),EPS(*),VDE(4,4),VPL(4,4),
1 DPL(4,4),SPL(4)
PI=4.D0*DATAN(1.D0)
C----- INITIALIZE MODEL PARAMETERS
COH=VPREE(NPRE+1)
PHI=VPREE(NPRE+2)*PI/180.D0
PSI=VPREE(NPRE+3)*PI/180.D0
C----- ELASTIC TRIAL
S1=SIGO(1)
S2=SIGO(4)
S3=SIGO(2)
S4=SIGO(3)
C----- FIND THE VALUE OF FUNCTION F ACCORDING TO THE ELASTIC TRIAL
CALL INVAR(S1,S2,S3,S4,PRESS,R3J2,VLODE)

```

```

ROOT3=DSQRT(3.D0)
SIN1=DSIN(PHI)
COS1=DCOS(PHI)
SIN2=DSIN(VLODE)
COS2=DCOS(VLODE)
FNEW=SIN1*PRESS+R3J2*(COS2/ROOT3-SIN2*SIN1/3.D0)-COH*COS1
IF(FNEW.LT.0.D0) GO TO 100
C----- FIND THE VALUE OF FUNCTION F ACCORDING TO PREVIOUS STRESSES
S1=SIG(1)
S2=SIG(4)
S3=SIG(2)
S4=SIG(3)
CALL INVAR(S1,S2,S3,S4,PRESS,R3J2,VLODE)
SIN2=DSIN(VLODE)
COS2=DCOS(VLODE)
F=SIN1*PRESS+R3J2*(COS2/ROOT3-SIN2*SIN1/3.D0)-COH*COS1
C----- OBTAIN THE STRESS-DEPENDENT PLASTIC MATRIX PL
CALL D02(VPREE,VDE,3,0)
C     S1=SIG0(1)
C     S2=SIG0(4)
C     S3=SIG0(2)
C     S4=SIG0(3)
CALL FMPLMC(S1,S2,S3,S4,VDE,VPL,PHI,PSI)
IF(F.GT.0.D0) FAC=1.D0
C----- COMPUTE THE SCALING FACTOR
IF(F.LE.0.D0) FAC=FNEW/(FNEW-F)
C----- OBTAIN ELASTIC-PLASTIC MATRIX DPL
DO 20 I=1,4
DO 20 J=1,4
20 DPL(I,J)=VDE(I,J)-FAC*VPL(I,J)
C----- COMPUTE THE ACTUAL STRESSES
CALL MULT(DPL,EPS,SPL,4,4,1)
DO 30 I=1,4
30 SIG(I)=SIG(I)+SPL(I)
MTYPE=2
RETURN
100 MTYPE=1
SIG(1)=SIG0(1)
SIG(2)=SIG0(2)
SIG(3)=SIG0(3)
SIG(4)=SIG0(4)
RETURN
END
SUBROUTINE FMPLMC(S1,S2,S3,S4,VDE,VPL,PHI,PSI)
C=====
C     TO FORM MATRIX PL FOR 2-D MOHR-COULOMB PERFECT PLASTICITY
C     INPUT      S1,S2,S3,S4,PHI,PSI AND MATRIX VDE
C     OUTPUT      MATRIX VPL (FULL)
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION VDE(4,4),VPL(4,4),DDQ(4),DDF(4),DQ(4),DF(4),DDQF(4,4)
C=====
CALL INVAR(S1,S2,S3,S4,PRESS,R3J2,THETA)
SRD=(2.D0*S1-S2-S3)/3.D0

```

```

STD=(2.D0*S2-S3-S1)/3.D0
SZD=(2.D0*S3-S1-S2)/3.D0
CALL QF(THETA,PHI,R3J2,F1,F2,F3)
CALL QF(THETA,PSI,R3J2,Q1,Q2,Q3)
CALL QSFS(F1,F2,F3,SRD,SZD,S4,STD,DF)
CALL QSFS(Q1,Q2,Q3,SRD,SZD,S4,STD,DQ)
CALL MULT(VDE,DF,DDF,4,4,1)
CALL MULT(VDE,DQ,DDQ,4,4,1)
VAL=0.DO
DO 10 I=1,4
10 VAL=VAL+DDF(I)*DQ(I)
DO 20 I=1,4
DO 20 J=1,4
DDQF(I,J)=DDQ(I)*DF(J)
20 DDQF(I,J)=DDQF(I,J)/VAL
CALL MULT(DDQF,VDE,VPL,4,4,4)
RETURN
END
SUBROUTINE INVAR(S1,S2,S3,S4,PRESS,R3J2,VLODE)
C=====
C COMPUTE INVARIANTS OF THE STRESS TENSOR
C=====
IMPLICIT REAL*8(A-H,O-Z)
R3J2=DSQRT(.5D0*((S1-S2)**2+(S2-S3)**2+(S3-S1)**2+6.D0*S4**2))
SRD=(2.D0*S1-S2-S3)/3.D0
STD=(2.D0*S2-S3-S1)/3.D0
SZD=(2.D0*S3-S1-S2)/3.D0
PRESS=(S1+S2+S3)/3.D0
SJ3=SRD*STD*SZD-STD*S4*S4
IF(R3J2.NE.0.DO) GO TO 10
VLODE=999.999D0
GO TO 20
10 VLODE=-13.5D0*SJ3/(R3J2*R3J2*R3J2)
PI=4.D0*DATAN(1.DO)
IF(VLODE.GE.1.DO) GO TO 30
IF(VLODE.LE.-1.DO) GO TO 40
ADJ=DSQRT(1.DO-VLODE*VLODE)
VLODE=-DATAN(VLODE/ADJ)/3.D0
GO TO 20
30 VLODE=PI/6.DO
GO TO 20
40 VLODE=-PI/6.DO
20 RETURN
END
SUBROUTINE QF(THETA,RAD,R3J2,X,Y,Z)
C=====
C TO OBTAIN (1) DERIVATIVES OF M-C YIELD FUNCTION WRT I1,J2,J3
C (2) DERIVATIVES OF M-C PLASTIC POTENTIAL WRT I1,J2,J3
C=====
IMPLICIT REAL*8(A-H,O-Z)
ROOT3=DSQRT(3.D0)
IF(DABS(THETA).LE..523424D0) GO TO 10
X=DSIN(RAD)
Y=.25D0/R3J2*(3.D0-DSIN(RAD))

```

```

Z=0.00
GO TO 20
10  X=DSIN(RAD)
    Y=DCOS(THETA)*ROOT3*.5D0/R3J2*((1.D0+DSIN(THETA)/DCOS(THETA)
    1*DSIN(3.D0*THETA)/DCOS(3.D0*THETA))+DSIN(RAD)/ROOT3
    2*(DSIN(3.D0*THETA)/DCOS(3.D0*THETA)-DSIN(THETA)/DCOS(THETA)))
    Z=(ROOT3*DSIN(THETA)+DSIN(RAD)*DCOS(THETA))*1.5D0
    1/(R3J2*R3J2*DCOS(3.D0*THETA))
20  RETURN
END
SUBROUTINE QSFS(A,B,C,R,Z,RZ,T,D)
C=====
C USE THE CHAIN RULE TO OBTAIN
C (1) DERIVATIVES OF M-C FAILURE FUNCTION WRT STRESS COMPONENTS
C (2) DERIVATIVES OF M-C PLASTIC POTENTIAL WRT STRESS COMPONENTS
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION D(4)
D(1)=A/3.D0+B*R+C/3.D0*(R*R+2.D0*Z*T+RZ*RZ)
D(2)=A/3.D0+B*Z+C/3.D0*(Z*Z+2.D0*R*T+RZ*RZ)
D(3)=2.D0*B*RZ-2.D0*C*RZ*T
D(4)=A/3.D0+B*T+C/3.D0*(2.D0*Z*R+T*T-2.D0*RZ*RZ)
RETURN
END
SUBROUTINE MULT(A,B,C,N1,N2,N3)
C=====
C MULTIPLY AN N1XN2 MATRIX A TIMES AN N2XN3 MATRIX B TO OBTAIN
C AN N1XN3 MATRIX C
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION A(N1,N2),B(N2,N3),C(N1,N3)
DO 10 I = 1,N1
DO 10 J = 1,N3
C(I,J)=0.0
DO 10 K = 1,N2
10  C(I,J)=C(I,J)+A(I,K)*B(K,J)
RETURN
END
SUBROUTINE DP2D(VPREE,SIG,EPS,NPRE,MTYPE)
C=====
C USE DRUCKER-PRAGER MODEL TO FIND ACTUAL STRESSES AT A G.P.
C STRESSES ARE AXISYMMETRIC
C MTYPE=0(TENSION CUTOFF),MTYPE=1(ELASTIC),MTYPE=2(FAILURE)
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION VPREE(*),SIG(*),EPS(*)
DATA CONV/.001D0/
C----- FUNCTION STATEMENT FOR EXPONENTIAL WITH SMALL ARGUMENT
C----- FAILURE ENVELOPE FUNCTION FOR SJ2
F(SJ1)=CB-CA*SJ1
C----- INITIALIZE MATERIAL PROPERTIES
PI=4.D0*DATAN(1.D0)
E=VPREE(1)
V=VPREE(2)

```

```

BULK=E/(3.DO-6.DO*V)
SHEAR=E/(2.DO+2.DO*V)
PHI=VPREE(NPRE+1)
CIN=VPREE(NPRE+2)
TANPHI=DTAN(PI*PHI/180.DO)
X=DSQRT(9.DO+12.DO*TANPHI**2)
CA=TANPHI/X
CB=3.DO*CIN/X
TCUT=VPREE(NPRE+3)
IF(CA.EQ.0.DO) FCUT=1.0D10
IF(CA.EQ.0.DO) GO TO 15
FCUT=CB/CA
C----- COMPUTE VOLUMETRIC STRAIN INCREMENT
15 DEV=EPS(1)+EPS(4)+EPS(2)
DEVO3=DEV/3.DO
DEXX=EPS(1)-DEVO3
DEYY=EPS(4)-DEVO3
C----- COMPUTE INITIAL MEAN NORMAL STRESS, STRESS DEVIATION TENSOR,
C----- AND STRESS INVARIANTS. NOTE THAT SZZ=-(SXX+SYY)
PRESS=(SIG(1)+SIG(4)+SIG(2))/3.DO
SXX=SIG(1)-PRESS
SYY=SIG(4)-PRESS
SXZ=SIG(3)
SJ1I=3.DO*PRESS
SJ2I=DSQRT(SXX*SXX+SYY*SYY+SXX*SYY+SXZ*SXZ)
C----- ELASTIC MATERIAL PROPERTIES
THREEK=3.DO*BULK
TWOG=2.DO*SHEAR
C----- ELASTIC TRIAL
SJ1=THREEK*DEV+SJ1I
SXX=SXX+TWOG*DEXX
SYY=SYY+TWOG*DEYY
SXZ=SXZ+SHEAR*EPS(3)
RATIO=1.0
MTYPE=1
C----- TENSILE CODING
TENCUT=DMIN1(FCUT,TCUT)
IF(SJ1.LT.TENCUT) GO TO 100
SJ1=TENCUT
MTYPE=0
RATIO=0.DO
GO TO 200
C----- CHECK FAILURE ENVELOPE
100 CONTINUE
SJ2=DSQRT(SXX*SXX+SYY*SYY+SXX*SYY+SXZ*SXZ)
FJ1=F(SJ1)
FF=SJ2-FJ1
IF(FF.LT.0.DO) GO TO 200
C----- YIELD SURFACE CALCULATION
MTYPE=2
DFDJ1=(F(SJ1+CONV*SJ2)-FJ1)/(CONV*SJ2)
DLAMD=FF/(3.DO*THREEK*DFDJ1**2+.5DO*TWOG)
DEVP=-3.DO*DFDJ1*DLAMD
SJ1E=SJ1

```

```

SJ1=SJ1-THREEK*DEVP
RATIO=F(SJ1)/SJ2
200 CONTINUE
C----- COMPUTE ACTUAL STRESSES
SXX=SXX*RATIO
SYY=SYY*RATIO
SXZ=SXZ*RATIO
PRESS=SJ1/3.D0
SIG(1)=SXX+PRESS
SIG(4)=SYY+PRESS
SIG(2)=PRESS-SXX-SYY
SIG(3)=SXZ
RETURN
END
SUBROUTINE CAPI(VPREE,NPREIEL,MS,MH)
C=====
C DEFINE THE INITIAL POSITION OF THE CAP FOR EACH GAUSS POINT
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION VPREE(*),VSE(81),SIG(4),XSTART(9)
DATA IPG/9/,INSE/36/
C----- INITIALIZE SOME MODEL PARAMETERS
AA=VPREE(NPRE+1)
AB=VPREE(NPRE+2)
AC=VPREE(NPRE+3)
THETA=VPREE(NPRE+4)
CR=VPREE(NPRE+6)
C----- READ ELEMENT STRESS DATA
READ(MS,REC=IEL) (VSE(I),I=1,INSE)
DO 60 IG=1,IPG
C----- NOTE CHANGE OF SIGN CONVENTION (TENSION-NEGATIVE)
SIG(1)=-VSE(IG*4-3)
SIG(2)=-VSE(IG*4-2)
SIG(3)=VSE(IG*4-1)
SIG(4)=-VSE(IG*4)
C----- COMPUTE INITIAL SJ1 AND SJ2
PRESS=(SIG(1)+SIG(2)+SIG(4))/3.D0
SXX=SIG(1)-PRESS
SYY=SIG(4)-PRESS
SXZ=SIG(3)
SJ1I=3.D0*PRESS
SJ2I=DSQRT(SXX*SXX+SYY*SYY+SXX*SYY+SXZ*SXZ)
X=0.ODO
C----- FIND INITIAL X BY NEWTON'S METHOD
DO 10 I=1,30
F=X+CR*(AA+THETA*X-AC*DEXP(-AB*X))-SJ1I
DF=1.ODO+CR*THETA+CR*AC*AB*DEXP(-AB*X)
X=X-F/DF
IF(DABS(F).LT.1.D-10) GO TO 20
10 CONTINUE
20 IF(SJ2I.EQ.0.0D0) GO TO 50
C----- ITERATE BY TRIAL AND ERROR
DX=X-SJ1I
K=1

```

```

DO 40 N=1,50
K=-K
DX=-0.5D0*DX
DO 30 M=1,2
X=X+DX
B=AA+THETA*X-AC*DEXP(-AB*X)
VR=(SJII-X)**2+CR*CR*SJ2I*SJ2I
VL=CR*CR*B*B
VF=VR-VL
IF(DABS(VF).LT.1.D-10) GO TO 50
IF(K.EQ.-1.AND.VF.LT.0.0D0) GO TO 40
IF(K.EQ.1.AND.VF.GT.0.0D0) GO TO 40
30 CONTINUE
40 CONTINUE
50 XSTART(IG)=-X
60 CONTINUE
CALL WRMH(MH,IEL,IPG,4,XSTART,XSTART)
RETURN
END

SUBROUTINE CAP2D(VPREE,SIGO,SIG,EPS,ASTART,EL,NPRE,MTYPE)
C=====
C USE SANDLER CAP MODEL TO FIND THE ACTUAL STRESSES AT A G.P.
C STRESSES ARE AXISYMMETRIC. GEOP=GEOSTATIC PRESSURE
C MTYPE=0(TENSION CUTOFF), =1(ELASTIC), =2(FAILURE), =3(CAP ACTION)
C IT=ITERATIONS FOR CAP, NOCON=1 INDICATES MAX ITERATIONS (NIT)
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION VPREE(*),SIGO(*),SIG(*),EPS(*)
DATA CONV/.00001D0/,NIT/100/,LTYPE/1/
C---- FUNCTION STATEMENT FOR EXPONENTIAL WITH SMALL ARGUMENT
C---- FAILURE ENVELOPE FUNCTION STATEMENT FOR SJ1
F(SJ1)=CA+THETA*SJ1-CC*DEXP(-CB*SJ1)
C---- CAP FUNCTION STATEMENTS (CAPL=BIGL(EL),XL=X(EL))
BIGL(EL)=DMAX1(ASTART,EL)
R(CAPL)=CR+0.D0*CAPL
X(EL)=DMAX1(0.D0,EL+R(BIGL(EL))*F(EL))
EVP(XL)=CW*(1.D0-DEXP(-CD*XL))
SJ2C(SJ1,XL,CAPL)=DSQRT((XL-CAPL)**2-(SJ1-CAPL)**2)/R(CAPL)
C---- ELASTIC MODULI FUNCTIONS (EV IS CURRENT VALUE OF EVP(XL))
BMOD(SJ1,EV)=BULK+0.D0*SJ1+0.D0*EV
SMOD(SJ2,EV)=SHEAR+0.D0*SJ2+0.D0*EV
C---- INITIALIZE MODEL PARAMETERS
E=VPREE(1)
V=VPREE(2)
BULK=E/(3.D0-6.D0*V)
SHEAR=E/(2.D0+2.D0*V)
CA=VPREE(NPRE+1)
CB=VPREE(NPRE+2)
CC=VPREE(NPRE+3)
THETA=VPREE(NPRE+4)
CD=VPREE(NPRE+5)
CR=VPREE(NPRE+6)
CW=VPREE(NPRE+7)
TCUT=VPREE(NPRE+8)

```

```

FCUT=DLOG(CA/CC)/CB
C FCUT=0.D0
C----- CHANGE OF SIGN CONVENTION (TENSION-NEGATIVE)
SIG(1)=-SIG(1)
SIG(2)=-SIG(2)
SIG(4)=-SIG(4)
SIG0(1)=-SIG0(1)
SIG0(2)=-SIG0(2)
SIG0(4)=-SIG0(4)
EPS(1)=-EPS(1)
EPS(2)=-EPS(2)
EPS(4)=-EPS(4)
ASTART=-ASTART
EL=-EL
C----- COMPUTE VOLUMETRIC STRAIN INCREMENT AND
C----- STRAIN DEVIATION INCREMENT TENSOR
DEV=EPS(1)+EPS(4)+EPS(2)
DEVO3=DEV/3.D0
DEXX=EPS(1)-DEVO3
DEYY=EPS(4)-DEVO3
C----- COMPUTE INITIAL MEAN NORMAL STRESS, STRESS DEVIATION TENSOR,
C----- AND STRESS INVARIANTS. NOTE THAT SZZ=-(SXX+SYY)
PRESS=(SIG(1)+SIG(4)+SIG(2))/3.D0
SXX=SIG(1)-PRESS
SYY=SIG(4)-PRESS
SXZ=SIG(3)
SJ1I=3.D0*PRESS
SJ2I=DSQRT(SXX*SXX+SYY*SYY+SXX*SYY+SXZ*SXZ)
CAPL=BIGL(EL)
XL=X(EL)
EVPI=EVP(XL)
C----- ELASTIC MATERIAL PROPERTIES
THREEK=3.D0*BMOD(SJ1I,EVPI)
TWOG=2.D0*SMOD(SJ2I,EVPI)
C----- ELASTIC TRIAL
SJ1=THREEK*DEV+SJ1I
SXX=SXX+TWOG*DEXX
SYY=SYY+TWOG*DEYY
SXZ=SXZ+TWOG*EPS(3)/2.D0
RATIO=1.D0
MTYPE=1
C----- TENSILE CODING - GEOP=GRAVITY STRESS
TENCUT=DMIN1(FCUT,TCUT)
IF(SJ1.GT.TENCUT) GO TO 20
SJ1=TENCUT
MTYPE=0
IF(LTYPE.EQ.2.OR.EL.LE.ASTART) GO TO 110
C----- TENSION DILANTANCY CODING
ELL=DMAX1(ASTART,EL+CONV*F(EL))
XLL=X(ELL)
DENOM=EVP(XLL)-EVPI
IF(DENOM.GT.0.D0) GO TO 10
EL=ASTART
GO TO 110

```

```

10    DEVP=DEV-(SJ1-SJ1I)/THREEK
      EL=EL+DEVP*(ELL-EL)/DENOM
      EL=DMIN1(START,EL)
      GO TO 110
C----- CHECK FAILURE ENVELOPE
20    CONTINUE
      SJ2=DSQRT(SXX*SXX+SYY*SYY+SXX*SYY+SXZ*SXZ)
      IF(SJ1.GT.CAPL) GO TO 50
      TMISES=SJ2C(CAPL,XL,CAPL)
      FJ1=F(SJ1)
      FF=SJ2-DMIN1(FJ1,TMISES)
      IF(FF.LT.0.D0) GO TO 110
C----- FAILURE SURFACE CALCULATION
      MTYPE=2
      DFDJ1=0.D0
      IF(FJ1.LT.TMISES) DFDJ1=(F(SJ1+CONV*SJ2)-FJ1)/(CONV*SJ2)
      DLAMD=FF/(3.D0*THREEK*DFDJ1**2+.5D0*TWOG)
      DEVP=-3.D0*DFDJ1*DLAMD
      SJ1E=SJ1
      SJ1=SJ1-THREEK*DEVP
C----- DILATANCY AND CORNER CODING
      IF(LTYPE.EQ.1.AND.EL.GT.START) GO TO 30
      SJ1=DMIN1(SJ1,CAPL)
      GO TO 40
30    CONTINUE
      ELL=DMIN1(SJ1E,EL+CONV*SJ2)
      XLL=X(ELL)
      DENOM=EVP(XLL)-EVPI
      IF(DENOM.GT.0.D0) GO TO 35
      SJ1=SJ1E
      EL=DMAX1(START,SJ1)
      GO TO 40
35    DILAT=(ELL-EL)/DENOM
      ELINT=EL
      EL=DMAX1(START,ELINT+DEVP*DILAT)
      IF(SJ1.LE.BIGL(EL)) GO TO 40
      SJ1=(DILAT*SJ1E+THREEK*ELINT)/(DILAT+THREEK)
      EL=DMAX1(START,SJ1)
40    CONTINUE
      FJ1=F(SJ1)
      RATIO=DMIN1(FJ1,TMISES)/SJ2
      GO TO 110
C----- CAP CALCULATION
50    CONTINUE
      IF(SJ1.GT.XL) GO TO 60
      IF(SJ2.LE.SJ2C(SJ1,XL,CAPL)) GO TO 110
60    CONTINUE
      SJ1E=SJ1
      SJ2E=SJ2
      ELL=EL
      ELR=SJ1E
      IF(SJ1E.GE.XL) FL=(EL-SJ1E)/(EL-XL)
      IF(SJ1E.LT.XL) FL=2.D0*SJ2E/(SJ2E+SJ2C(SJ1E,XL,CAPL))-1.D0
      XR=X(ELR)

```

```

SJ1R=SJ1E-THREEK*(EV(P(XR))-EVPI)
FR=(XR-SJ1R)/(ELR-XR)
COMP=CONV*F((FL*XR-FR*XL)/(FL-FR))
IF(DABS(SJ1)+SJ2.LT.COPM) GO TO 110
MTYPE=3
FOLD=0.DO
DO 90 IT=1,NIT
EL=(FL*ELR-FR*ELL)/(FL-FR)
XL=X(EL)
DEVP=EV(P(XL)-EVPI
SJ1=SJ1E-THREEK*DEVP
CAPL=BIGL(EL)
IF(SJ1.GE.XL) FC=(EL-SJ1)/(EL-XL)
IF(SJ1.LE.CAPL) FC=(XL-SJ1)/(CAPL-XL)
IF(SJ1.GE.XL.OR.SJ1.LE.CAPL) GO TO 70
SJ2=SJ2C(SJ1,XL,CAPL)
DELJ1=CONV*(XL-SJ1)
DESP=0.DO
IF(DELJ1) 65,66,65
65 DESP=(DEVP/6.DO)*(DELJ1/(SJ2-SJ2C(SJ1+DELJ1,XL,CAPL)))
CONTINUE
SJ2TRY=SJ2+TWOG*DESP
FC=(SJ2E-SJ2TRY)/(SJ2E+SJ2TRY)
IF(DABS(SJ2E-SJ2TRY).LE.COMP) GO TO 100
IF(FC.GT.0.DO.AND.(SJ1-CAPL).LE.DELJ1) GO TO 100
70 IF(FC.GT.0.DO) GO TO 80
ELR=EL
FR=FC
IF(FOLD.LT.0.DO) FL=.5DO*FL
GO TO 90
80 CONTINUE
ELL=EL
FL=FC
IF(FOLD.GT.0.DO) FR=.5DO*FR
90 FOLD=FC
NOCON=1
SJ1=DMIN1(SJ1,XL)
IF(SJ1.LT.BIGL(ELR)) SJ1=CAPL
SJ2=DMIN1(SJ2E,SJ2C(SJ1,XL,CAPL))
100 RATIO=0.DO
IF(SJ2E.NE.0.DO) RATIO=SJ2/SJ2E
110 CONTINUE
SXX=SXX*RATIO
SYY=SYY*RATIO
SXZ=SXZ*RATIO
PRESS=SJ1/3.DO
SIG(1)=-(SXX+PRESS)
SIG(4)=-(SYY+PRESS)
SIG(2)=-(PRESS-SXX-SYY)
SIG(3)=SXZ
EPS(1)=-EPS(1)
EPS(2)=-EPS(2)
EPS(4)=-EPS(4)
ASTART=-ASTART

```

```

EL=-EL
RETURN
END
SUBROUTINE VSHEAR(VPREE,VCORE,SIG,EPS,ROWA,ATMP,NPRE,MTYPE)
C=====
C   VSHEAR FINDS THE TANGENT SHEAR MODULUS ACCORDING TO
C   A VARIABLE MODULI MODEL BASED ON THE HYPERBOLIC EQUATION
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION VPREE(*),VCORE(*),SIG(*),EPS(*)
C-----
PI=4.0*DATAN(1.0)
MTYPE=1
C----- INITIALIZE MODEL PARAMETERS
E=VPREE(1)
V=VPREE(2)
G=VPREE(3)
PHI=VPREE(NPRE+1)*PI/180.0
COH=VPREE(NPRE+2)
XK=VPREE(NPRE+3)
EXPN=VPREE(NPRE+4)
RF=VPREE(NPRE+5)
C----- FIND THE NORMAL STRESS ANGLE WITH RESPECT TO X-AXIS
DX13=VCORE(5)-VCORE(1)
DY13=VCORE(6)-VCORE(2)
DL13=DSQRT(DX13**2+DY13**2)
DX17=VCORE(13)-VCORE(1)
DY17=VCORE(14)-VCORE(2)
DL17=DSQRT(DX17**2+DY17**2)
IF(DL13.LT.DL17) COS=DX13/DL13
IF(DL13.LT.DL17) SIN=DY13/DL13
IF(DL13.GT.DL17) COS=DX17/DL17
IF(DL13.GT.DL17) SIN=DY17/DL17
C----- INCREMENT THE STRAIN COMPONENTS
20 DEPSX=EPS(1)/20.0D0
DEPSY=EPS(2)/20.0D0
DEPSXY=EPS(3)/20.0D0
D11=E*(1.0D0-V)/((1.0D0+V)*(1.0D0-2.0D0*V))
D12=D11*V/(1.0D0-V)
C----- ITERATE 20 TIMES TO OBTAIN APPROXIMATE ACTUAL STRESSES
DO 100 I=1,20
D33=G
C----- COMPUTE THE NORMAL STRESSES. NOTE CHANGE OF SIGN
SN=SIG(1)*COS*COS+SIG(2)*SIN*SIN+2.0D0*SIN*COS
SS=SIG(3)*(COS*COS-SIN*SIN)-(SIG(1)-SIG(2))*SIN*COS
SS=DABS(SS)
IF(SN.LT.1.0D-30) GO TO 30
C----- COMPUTE THE TAGENT MODULUS
FACTOR=1.0D0-RF*SS/(COH+SN*DTAN(PHI))
IF(FACTOR.LT.1.0D-10) GO TO 30
D33=XK*ROWA*(SN/ATMP)**EXPN*FACTOR*FACTOR
IF(D33.LE.0.0D0) GO TO 30
MTYPE=2
30 SIG(1)=SIG(1)+D11*DEPSX+D12*DEPSY

```

```

      SIG(2)=SIG(2)+D12*DEPSX+D11*DEPSY
      SIG(3)=SIG(3)+D33*DEPSXY
100  CONTINUE
      RETURN
      END
      SUBROUTINE FVM2D(VPREE,SIG,VDE,ATMP,IKEL,NPRE)
C=====
C   FVM2D FINDS THE TANGENT STRESS-STRAIN MATRIX ACCORDING TO
C   A VARIABLE MODULUS MODEL BASED ON THE HYPERBOLIC EQUATION
C=====
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION VPREE(*),SIG(4),VDE(4,4)
C-----
C----- STATEMENT FUNCTIONS
      EI(A,B,C,D)=A*B*(C/B)**D
      S(A,B,C,D)=(A-B)*(1.D0-DSIN(C))/(2.D0*(D*DCOS(C)+B*DSIN(C)))
C----- INITIALIZE MODEL PARAMETERS
      PHI0=VPREE(NPRE+1)
      DPHI=VPREE(NPRE+2)
      COH=VPREE(NPRE+3)
      XK=VPREE(NPRE+4)
      EXPN=VPREE(NPRE+5)
      RF=VPREE(NPRE+6)
      XKB=VPREE(NPRE+7)
      EXPM=VPREE(NPRE+8)
      XKUR=VPREE(NPRE+9)
C----- OBTAIN PRINCIPAL STRESSES. NOTE CHANGE OF SIGN CONVENTIONS
C----- TENSION - NEGATIVE, COMPRESSION - POSITIVE
      PI=4.D0*Datan(1.D0)
      C1=(-SIG(1)-SIG(2))/2.D0
      C2=(SIG(2)-SIG(1))/2.D0
      TAUMAX=DSQRT(C2**2+SIG(3)**2)
      SIG1=C1+TAUMAX
      SIG3=C1-TAUMAX
      IF(SIG1.LT.1.D-20.OR.SIG3.LT.1.D-20) GO TO 10
      PHI=PHI0-DPHI*DLOG10(SIG3/ATMP)
      PHI=PHI*PI/180.D0
      SR=S(SIG1,SIG3,PHI,COH)
      ET=EI(XK,ATMP,SIG3,EXPN)*(1.D0-RF*SR)**2
      BULK=EI(XKB,ATMP,SIG3,EXPM)
      VT=.5D0-ET/BULK/6.D0
      IF(VT.LT..2D0) VT=.2D0
      VPREE(1)=ET
      VPREE(2)=VT
10    CALL D02(VPREE,VDE,IKEL,0)
      RETURN
      END
      SUBROUTINE FMC2D (VPREE,SIG,VDE,IKEL,NPRE,MTYPE)
C=====
C   FMC2D YIELDS THE 2-D ELASTO-PLASTIC MATRIX ACCORDING TO THE
C   MOHR-COULOMB FORMULATION (PLANE STRAIN)
C=====
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION VPREE(*),SIG(4),VDE(4,4),VPL(4,4)

```

```

C-----
      IF(MTYPE.EQ.1) GO TO 20
      PI=4.D0*DATAN(1.D0)
C----- INITIALIZE MODEL PARAMETERS
      COH=VPREE(NPRE+1)
      PHI=VPREE(NPRE+2)*PI/180.D0
      PSI=VPREE(NPRE+3)*PI/180.D0
      S1=SIG(1)
      S2=SIG(4)
      S3=SIG(2)
      S4=SIG(3)
C----- OBTAIN STRESS-DEPENDENT PLASTIC MATRIX
      CALL D02(VPREE,VDE,IKEL,0)
      CALL FMPLMC(S1,S2,S3,S4,VDE,VPL,PHI,PSI)
C----- OBTAIN ELASTO-PLASTIC MATRIX
      DO 10 I=1,4
      DO 10 J=1,4
10   VDE(I,J)=VDE(I,J)-VPL(I,J)
      RETURN
20   CALL D02(VPREE,VDE,IKEL,0)
      RETURN
      END
      SUBROUTINE FDP2D (VPREE,SIG,VDE,IKEL,NPRE,MTYPE)
C=====
C     FDP2D YIELDS THE 2-D ELASTO-PLASTIC STRESS-STRAIN MATRIX
C     ACCORDING TO DRUCKER-PRAGER FORMULATION
C     MTYPE .LE. 1    FORM ELASTIC MATRIX
C     .EQ. 2    FORM ELASTO-PLASTIC MATRIX
C=====
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION VPREE(*),SIG(4),VDE(4,4)
C-----
      IF(MTYPE.LE.1) GO TO 10
      PI=4.D0*DATAN(1.D0)
C----- INITIALIZE MATERIAL PROPERTIES
      E=VPREE(1)
      V=VPREE(2)
      PHI=VPREE(NPRE+1)
      CIN=VPREE(NPRE+2)
      TANPHI=DTAN(PI*PHI/180.D0)
      X=DSQRT(9.D0+12.D0*TANPHI**2)
      ALP=TANPHI/X
      ALK=3.D0*CIN/X
C----- AI1C IS THE FIRST INVARIANT OF STRESS
C----- AJ2C IS THE SQUARE ROOT OF THE SECOND INVARIANT
      AI1C=SIG(1)+SIG(4)+SIG(2)
      AJ2C=DSQRT(((SIG(1)-SIG(2))**2+(SIG(2)-SIG(4))**2+(SIG(4)-
1 SIG(1))**2)/6.D0+SIG(3)**2)
      ABK=E/(3.D0-6.D0*V)
      G=E/(2.D0+2.D0*V)
      AHH=3.D0*ABK*ALP/(2.D0*G)-AI1C/(6.D0*AJ2C)
      PP=AJ2C*(1.D0+9.D0*ALP*ALP*ABK/G)
      CC=1.D0/(2.D0*PP*AJ2C)
      AA=AHH/PP

```

```

BB=2.D0*AHH*AHH/(1.D0+9.D0*ALP*ALP*ABK/G)-3.D0*V*ABK/E
G2=2.D0*G
VDE(1,1)=G2*(1.D0-2.D0*AA*SIG(1)-BB-CC*SIG(1)*SIG(1))
VDE(1,3)=G2*(-AA*SIG(3)-CC*SIG(3)*SIG(1))
VDE(1,2)=G2*(-AA*(SIG(1)+SIG(2))-BB-CC*SIG(1)*SIG(2))
VDE(2,2)=G2*(1.D0-2.D0*AA*SIG(2)-BB-CC*SIG(2)*SIG(2))
VDE(2,3)=G2*(-AA*SIG(3)-CC*SIG(3)*SIG(2))
VDE(3,3)=G2*(0.5D0-CC*SIG(3)*SIG(3))
VDE(2,1)=VDE(1,2)
VDE(3,1)=VDE(1,3)
VDE(3,2)=VDE(2,3)
VDE(1,4)=G2*(-AA*(SIG(1)+SIG(4))-BB-CC*SIG(1)*SIG(4))
VDE(2,4)=G2*(-AA*(SIG(2)+SIG(4))-BB-CC*SIG(2)*SIG(4))
VDE(3,4)=G2*(-AA*SIG(3)-CC*SIG(3)*SIG(4))
VDE(4,1)=VDE(1,4)
VDE(4,2)=VDE(2,4)
VDE(4,3)=VDE(3,4)
VDE(4,4)=G2*(1.D0-2.D0*AA*SIG(4)-BB-CC*SIG(4)*SIG(4))
RETURN
10 CALL D02(VPREE,VDE,IKEL,0)
RETURN
END
SUBROUTINE FCP2D(VPREE,SIG,C,EL,IKEL,NPRE,MTYPE)
C=====
C   FCP2D YIELDS THE 2-D ELASTO-PLASTIC MATRIX ACCORDING TO
C   THE SANDLER CAP MODEL
C   MTYPE=0(TENSION CUTOFF), =1(ELASTIC), =2(FAILURE), =4(CAP ACTION)
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION VPREE(*),SIG(4),C(4,4)
C-----
C----- STATEMENT FUNCNTION FOR EXPONENTIAL WITH SMALL ARGUMENT
EXPS(Z)=DEXP(DMAX1(-500.D0,Z))
C----- FAILURE ENVELOPE FUNCTION FOR SJ2
F(SJ1)=CA-THETA*SJ1-CC*EXPS(CB*SJ1)
C----- CAP STATEMENT FUNCTIONS (CAPL=BIGL(EL),XL=X(EL))
R(CAPL)=CR+O.D0*CAPL
X(EL)=EL-R(EL)*F(EL)
SJ2C(SJ1,XL,CAPL)=DSQRT((XL-CAPL)**2-(SJ1-CAPL)**2)/R(CAPL)
IF(MTYPE.LE.1) GO TO 30
C----- INITIALIZE MODEL PARAMETERS
CA=VPREE(NPRE+1)
CB=VPREE(NPRE+2)
CC=VPREE(NPRE+3)
THETA=VPREE(NPRE+4)
CD=VPREE(NPRE+5)
CR=VPREE(NPRE+6)
CW=VPREE(NPRE+7)
C----- MEAN NORMAL STRESS, STRESS DEVIATION TENSOR, INVARIANTS
PRESS=(SIG(1)+SIG(4)+SIG(2))/3.D0
SXX=SIG(1)-PRESS
SYY=SIG(4)-PRESS
SXZ=SIG(3)
SJ1=3.D0*PRESS

```

```

SJ2=DSQRT(SXX*SXX+SYY*SYY+SXX*SYY+SXZ*SXZ)
CAPL=EL
XL=X(EL)
C----- BEGIN CALCULATION OF PROPER CEP
CALL D02(VPREE,C,IKEL,0)
C11=C(1,1)
C22=C(2,2)
C44=C(3,3)
C12=C(1,2)
C33=C(4,4)
C13=C(1,4)
C23=C(2,4)
C----- NOW DEFINE PROPER CEP FACTORS
IF(MTYPE.EQ.3) GO TO 10
C----- FAILURE SURFACE CEP FACTORS
BCEJ=-THETA-CB*CC*EXPS(CB*SJ1)
SJ26=6.D0*SJ2
DF1=BCEJ-(2.D0*SIG(1)-SIG(4)-SIG(2))/SJ26
DF2=BCEJ-(2.D0*SIG(2)-SIG(4)-SIG(1))/SJ26
DF3=BCEJ-(2.D0*SIG(4)-SIG(1)-SIG(2))/SJ26
DF4=-SXZ/SJ2
DNOM=0.D0
GO TO 20
C----- CAP CEP FACTORS
10 CONTINUE
FF=CR*DSQRT((XL-CAPL)**2-(SJ1-CAPL)**2)
SJKRY=-(SJ1-CAPL)/FF
SJ26=6.D0*SJ2
DF1=-(2.D0*SIG(1)-SIG(2)-SIG(4))/SJ26+SJKRY
DF2=-(2.D0*SIG(2)-SIG(1)-SIG(4))/SJ26+SJKRY
DF3=-(2.D0*SIG(4)-SIG(2)-SIG(1))/SJ26+SJKRY
DF4=-SXZ/SJ2
F1=1.D0+CR*THETA+CR*CC*CB*EXPS(CB*CAPL)
F2=CN*CD*EXPS(CD*XL)
DNOM=(DF1+DF2+DF3)*(XL-CAPL+(SJ1-XL)/F1)/F2/FF
C----- NOW CACULATE THE CEP ENTRIES
20 DNOM=-DNOM+C11*DF1*DF1+2.D0*C12*DF1*DF2+2.D0*C13*DF1*DF3+
1 C22*DF2*DF2+2.D0*C23*DF2*DF3+C33*DF3*DF3+C44*DF4*DF4
A1=C11*DF1+C12*DF2+C13*DF3
A2=C12*DF1+C22*DF2+C23*DF3
A3=C13*DF1+C23*DF2+C33*DF3
C(1,1)=C(1,1)-A1*A1/DNOM
C(2,2)=C(2,2)-A2*A2/DNOM
C(3,3)=C(3,3)-(C44*DF4)**2/DNOM
C(1,2)=C(1,2)-A1*A2/DNOM
C(2,1)=C(1,2)
C(1,3)=C(1,3)-C44*DF4*A1/DNOM
C(3,1)=C(1,3)
C(2,3)=C(2,3)-C44*DF4*A2/DNOM
C(3,2)=C(2,3)
C(4,4)=C(4,4)-A3*A3/DNOM
C(3,4)=C(3,4)-C44*DF4*A3/DNOM
C(4,3)=C(3,4)
C(1,4)=C(1,4)-A1*A3/DNOM

```

```

C(4,1)=C(1,4)
C(2,4)=C(2,4)-A2*A3/DNOM
C(4,2)=C(2,4)
RETURN
30 CALL D02(VPREE,C,IKEL,0)
RETURN
END
SUBROUTINE FVS2D(VPREE,VCORE,SIG,VDE,ROWA,ATMP,NPRE)
C=====
C   FVS2D YIELDS THE TANGENT MODULUS FOR NONLINEAR INTEFACE ELEMENT
C   WITH A VARIABLE MODULI MODEL BASED ON HYPERBOLIC EQUATION
C=====
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION VPREE(*),VCORE(*),SIG(4),VDE(4,4)
C-----
PI=4.D0*DATAN(1.D0)
CALL D02(VPREE,VDE,2,1)
C----- INITIALIZE MODEL PARAMETERS
PHI=VPREE(NPRE+1)*PI/180.D0
COH=VPREE(NPRE+2)
XK=VPREE(NPRE+3)
EXPN=VPREE(NPRE+4)
RF=VPREE(NPRE+5)
C----- FIND THE NORMAL STRESS ANGLE WITH RESPECT TO X-AXIS
DX13=VCORE(5)-VCORE(1)
DY13=VCORE(6)-VCORE(2)
DL13=DSQRT(DX13**2+DY13**2)
DX17=VCORE(13)-VCORE(1)
DY17=VCORE(14)-VCORE(2)
DL17=DSQRT(DX17**2+DY17**2)
IF(DL13.LT.DL17) COS=DX13/DL13
IF(DL13.LT.DL17) SIN=DY13/DL13
IF(DL13.GT.DL17) COS=DX17/DL17
IF(DL13.GT.DL17) SIN=DY17/DL17
C----- COMPUTE THE NORMAL STRESSES. NOTE CHANGE OF SIGN
SN=SIG(1)*COS*COS+SIG(2)*SIN*SIN+2.D0*SIN*COS
SS=SIG(3)*(COS*COS-SIN*SIN)-(SIG(1)-SIG(2))*SIN*COS
SS=DABS(SS)
IF(SN.LT.1.D-30) RETURN
C----- COMPUTE THE TAGENT MODULUS
FACTOR=1.D0-RF*SS/(COH+SN*DTAN(PHI))
IF(FACTOR.LT.1.D-10) RETURN
D33=XK*ROWA*(SN/ATMP)**EXPN*FACTOR*FACTOR
IF(D33.LE.0.D0) RETURN
VDE(3,3)=D33
RETURN
END

```

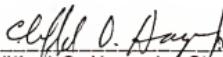
BIOGRAPHICAL SKETCH

Hsi Chi Yang was born in Taichung, Taiwan, on April 5, 1953. He graduated from Taichung First High School in Taichung, Taiwan, in June of 1968. He received his Bachelor of Science in Architecture degree from Chung Yuan University in Tao Yuan, Taiwan, in June of 1976.

After working two years in the architectural industry in Taiwan, he enrolled in the graduate program in the College of Architecture at University of Florida where he was awarded his Master of Arts in Architecture and Master of Building Construction degrees in 1980 and 1982, respectively. Since September of 1982 he has been a doctoral candidate in the Civil Engineering Department at the University of Florida.

Mr. Yang married Ye-Chien Chen on August 31st, 1979, and they have one son, Louis Yang, age seven.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



Clifford O. Hays, Jr., Chairperson
Professor of
Civil Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



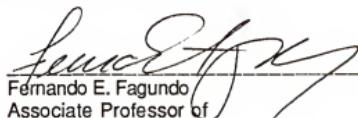
Frank C. Townsend
Professor of
Civil Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



Lawrence E. Malvern
Professor of
Aerospace Engineering, Mechanics, and
Engineering Science

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



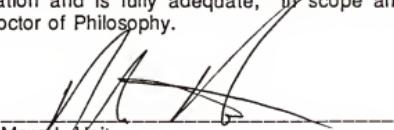
Fernando E. Fagundo
Associate Professor of
Civil Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



Michael C. McVay
Associate Professor of
Civil Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



Marc I. Hoit
Associate Professor of
Civil Engineering

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

August 1988



Herbert A. Bevis
Dean, College of Engineering

Dean, Graduate School